KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Security and Privacy of Implantable Medical Devices

Eduard Marin

Supervisors:
Prof. dr. ir. Bart Preneel
Dr. ir. Dave Singelée

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

March 2018

Security and Privacy of Implantable Medical Devices

Eduard MARIN

Examination committee:
Prof. dr. Hilde Heynen, chair
Prof. dr. ir. Bart Preneel, supervisor
Dr. ir. Dave Singelée, supervisor
Prof. dr. ir. Vincent Rijmen
Prof. dr. ir. Christophe Huygens
Prof. dr. ir. Danny Hughes
Prof. dr. Flavio D. Garcia
(University of Birmingham)
Prof. dr. Mauro Conti
(University of Padua)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

March 2018

Acknowledgements

"Si vols viure el que somies
has de començar somiant"
Txarango

This thesis is the result of a long journey which would not have been possible without the help and support of many people.

Foremost, I want to thank my promoter, Prof. Bart Preneel, for giving me the opportunity to pursue a PhD at COSIC and for providing such an excellent environment to conduct research. I also thank Bart for the trust he showed me, for the freedom he has given me and for his insightful comments which helped me to improve my work. In the same way, I want to thank my co-promoter, Dr. Dave Singelée, for being the first to believe that I could do a PhD and for playing such an important role not only in my doctoral studies, but also in my future professional career. I feel very fortunate to have had him as a supervisor and to have worked with him.

I would like to extend my gratitude to my assessors, Prof. Vincent Rijmen, Prof. Christophe Huygens and Prof. Flavio Garcia, for their comments and remarks throughout my PhD. I am happy that Prof. Danny Hughes and Prof. Mauro Conti accepted to be in my examination committee and I thank them for carefully reading my thesis and providing valuable feedback. I also thank Prof. Hilde Heynen for chairing the jury.

During my stay at COSIC, I have had the opportunity to write papers together with very talented researchers. My special thanks go to all my co-authors: Dave Singelée, Bohan Yang, Mustafa A. Mustafa, Enrique Argones Rúa, Aysajan Abidin, Flavio Garcia, Tom Chothia, Pieter Robyns and Vladimir Volski. I enjoyed a lot working with you and learnt many things from all of you.

I wish to thank Prof. Rik Willems, Stefan Foulon, Prof. Bart Nuttin, John Das, Prof. Pieter Gillard and Saskia Vanderwegen for helping and providing us with all the necessary medical equipment to conduct this project. I also thank Prof. Rik Willems and Prof. Bart Nuttin for letting me into the operating room. I will never forget this experience.

COSIC was (and will always be) a very nice place to work. I would like to thank all my former and current colleagues for making my stay at COSIC so enjoyable. A big thanks goes to those with whom I had lunch in Alma, had coffees, went karting and went for dinner together. A special mention goes to Dave, Dana and Yuan, for their friendship and for being such nice office mates. This thesis would not be the same without the amazing abstract translation that Jan Pieter and Lennert made to Dutch. Péla Noë was the best neighbour that I could have in COSIC. She was always very friendly with me and was always willing to help me with all the administrative problems I had. I also thank Wim, John, Elsy and Saartje for being so nice and making my life in COSIC easier.

I want to thank Txarango for turning the rainy, cloudy days into sunny, clear days. Your songs and your lyrics have always been a source of inspiration for me.

My friends have always been very supportive and have helped me to disconnect from work when needed. You know who you are and you also know that it is difficult to mention all of you here. But a very special thanks goes to Rosa Pedrero, Monica Vara, Aniol Saiz, Victor Bravo, Maria Bogaerts, Rita Faria, Toni Perez, Sofia Caubet, Blanca Santa Cecilia, Javi Rodriguez, Virginia Espinosa, Marta Farré, Itziar Muiños, Javi Cañas, Cristina Puñal, Myriam Verdonk, Anna Vila, Rosa Carmeno and Jonathan Jimenez.

I wish to thank my very good friend Manel Cuadros, who I miss very much since he left us, for his real friendship and his support. Manel was a happy, strong, brave guy from whom I learnt a lot. Although you could not come to visit me in Leuven, you have always been with me in the office.

Last but not least, I would like to devote some words to my parents, Francesc and Rosa, my brother, Ricard, and my dog, Kiri. There are not enough words to express how much I love you. Even though I live far away, I have always felt extremely close to you. I would like to thank you for your unconditional support, for always being by my side in the good and bad moments and especially for living this adventure first-hand. You know that everything achieved so far is as mine as yours. This thesis is dedicated to you.

Que trobis tot allò que busques
Que et sigui lleu la direcció
I vinguin plens de melodies els nous dies
que siguis fort. Que tot et vagi bé!

Eduard Marin
Leuven, March 2018

Abstract

This thesis deals with the security and privacy of Implantable Medical Devices (IMDs). Specifically, we analyse the security of widely used IMDs, and propose practical and effective countermeasures to address the security issues we have identified.

We first propose a protocol that allows an IMD to establish an end-to-end secure channel with a hospital while preserving the patient's privacy. This enables remote monitoring and reprogramming of the patient's IMD through a base station installed in the patient's home. Our solution prevents unauthorised entities and adversaries from learning to whom the data belongs and to which hospital the medical data is sent, among others. We also present a key establishment protocol through which the base station and the IMD can agree on a symmetric session key without needing to share any prior secrets. These goals are achieved by using a physiological signal extracted from the patient's body in combination with fuzzy extractors.

Next, we perform a security analysis of an insulin pump system and we present various attacks. Furthermore, we study the feasibility of using cryptography to protect the wireless communication between the insulin pump and the remote control. To this end, we present a cryptographic AES-based solution with an updated message format that is optimised for energy consumption. We propose multiple alternatives of our solution and implement them in an openMSP430, a 16-bit micro-controller similar to the one used in the insulin pump. For each of these alternatives, we measure the extra computation and communication energy cost due to the use of cryptography both in the remote control and the insulin pump. Finally, we identify possible ways of decreasing the communication cost.

To the best of our knowledge, we are the first to document the reverse engineering and security analysis of the proprietary protocol between a device programmer and some of the latest generation of Implantable Cardioverter Defibrillators (ICDs) from one of the leading manufacturers over a long-range wireless channel.

Our goal is to evaluate the feasibility of reverse-engineering the proprietary protocol by adversaries with limited resources who do not have physical access to the devices but can only eavesdrop on the wireless communications. Our work reveals the first attempt – known by the scientific community – to obfuscate the data that is transmitted over the air. In addition, we demonstrate attacks that can compromise the ICD's availability and the patient's privacy, and give evidence that replay and spoofing attacks are also possible. All our findings apply to at least 10 types of ICDs that are currently on the market. We also discuss several ways of how adversaries can bypass the activation procedure – which requires to be in close proximity to the patient – to send maliciously crafted commands to the ICD from several meters away. Finally, several short-term and long-term countermeasures are proposed, including a novel semi-offline key agreement protocol that we formally verify using ProVerif.

Furthermore, we describe the process of how to reverse engineer the proprietary protocol between a device programmer and a neurostimulator to communicate over a short-range channel. Subsequently, we assess the feasibility and conduct several types of attacks on neurostimulators. To preclude these attacks, we present a complete security architecture that includes the generation and transportation of keys from the neurostimulator to the device programmer and the necessary cryptographic protocols to secure the communication flow. For generating the key on the neurostimulator, we investigate the potential of using a signal extracted from the patient's brain as a source of randomness. We also propose a novel technique for securely and reliably transporting the key from the neurostimulator to the device programmer. Our technique leverages the fact that both the patient's skin and the neurostimulator's case are conductive.

To conclude, we provide a critical evaluation of countermeasures that rely on patient's physiological signals for establishing a cryptographic key between two devices. Our work reveals serious security weaknesses in two pairing protocols proposed in the literature. Furthermore, we show that most of the existing countermeasures rely on unrealistic assumptions that underestimate the adversaries' capabilities. This work concludes by providing a set of recommendations on how to securely use physiological signals in cryptographic protocols.

Beknopte samenvatting

Dit proefschrift behandelt de beveiliging en privacy van *Implantable Medical Devices (IMDs)*. Concreet analyseren we de beveiliging van veelgebruikte IMD's en stellen we praktische en effectieve tegenmaatregelen voor om de beveiligingsproblemen aan te pakken die we hebben vastgesteld.

We stellen eerst een protocol voor waarmee een IMD een *end-to-end* beveiligd kanaal met een ziekenhuis kan opzetten met behoud van de privacy van de patiënt. Dit maakt monitoring op afstand en herprogrammering van de IMD van de patiënt mogelijk via een basisstation dat bij de patiënt thuis is geïnstalleerd. Onze oplossing voorkomt onder andere dat ongeautoriseerde entiteiten en tegenstanders leren van wie de gegevens zijn en naar welk ziekenhuis de medische gegevens worden verzonden. We presenteren ook een nieuw sleuteluitwisselingsprotocol waarmee het basisstation en de IMD overeenstemming kunnen bereiken over een symmetrische sessiesleutel zonder voorafgaand geheimen te hoeven delen. Deze doelen worden bereikt door een fysiologisch signaal te gebruiken dat geëxtraheerd wordt uit het lichaam van de patiënt, in combinatie met *fuzzy extractors*.

Vervolgens voeren we een beveiligingsanalyse uit van een insulinepompsysteem en presenteren we verschillende aanvallen. Verder bestuderen we de haalbaarheid van het gebruik van cryptografie om de draadloze communicatie tussen de insulinepomp en een bijbehorende afstandsbediening te beschermen. Hiertoe presenteren we een op AES gebaseerde cryptografische oplossing met een geüpdatete berichtindeling die is geoptimaliseerd voor energieverbruik. We stellen meerdere alternatieven van onze oplossing voor en implementeren deze in een openMSP430, een 16-bits microcontroller die gelijkaardig is aan die van de insulinepomp. Voor elk van deze alternatieven meten we hoeveel energie er extra verbruikt wordt in zowel de afstandsbediening als de insulinepomp voor berekeningen en communicatie. Ten slotte identificeren we mogelijke manieren om de energieconsumptie van de communicatie te verlagen.

Vervolgens voeren we de eerste beveiligingsanalyse en *reverse engineering* uit van het gepatenteerde protocol tussen een apparaatprogrammeur en enkele van de nieuwste generatie *Implantable Cardioverter Defibrillators (ICDs)* via een langeafstands draadloos kanaal. Ons doel is om de haalbaarheid te evalueren van het *reverse-engineer* van het gepatenteerde protocol door tegenstanders met beperkte middelen die geen fysieke toegang tot de apparaten hebben maar alleen de berichten kunnen onderscheppen die draadloos worden verzonden. Daarnaast demonstreren we aanvallen die de beschikbaarheid van de ICD en de privacy van de patiënt in gevaar kunnen brengen, en kunnen we aantonen dat herhaling en spoofing-aanvallen ook mogelijk zijn. Al onze bevindingen zijn van toepassing op ten minste 10 soorten ICD's die momenteel op de markt zijn. We bespreken ook verschillende manieren waarop tegenstanders de activeringsprocedure kunnen omzeilen - waarvoor men zich normaal dicht bij de patiënt moet bevinden - om vanaf enkele meters schadelijke berichten naar de ICD te sturen. Ten slotte worden verschillende kortetermijn en langetermijn tegenmaatregelen voorgesteld om de bestaande kwetsbaarheden te reduceren of op te lossen, waaronder een nieuw semi-offline sleutelovereenkomstprotocol dat we formeel verifiëren met behulp van ProVerif.

Verder beschrijven we het proces van het *reverse engineer* van het gepatenteerde protocol tussen een apparaatprogrammeur en een neurostimulator om te communiceren over een korteafstandskanaal. Vervolgens beoordelen we de haalbaarheid en voeren we verschillende soorten aanvallen uit op neurostimulatoren. Om deze aanvallen te verhinderen, presenteren we een complete beveiligingsarchitectuur die het genereren en transporteren van sleutels van de neurostimulator naar de apparaatprogrammeur mogelijk maakt en de noodzakelijke cryptografische protocollen omvat om de communicatiestroom te beveiligen. Voor het genereren van de sleutel op de neurostimulator, onderzoeken we het potentieel van het gebruik van een signaal uit de hersenen van de patiënt als een bron van willekeur. We stellen ook een nieuwe techniek voor om de sleutel veilig en betrouwbaar van de neurostimulator naar de apparaatprogrammeur te transporteren, waarbij gebruik wordt gemaakt van het feit dat zowel de huid van de patiënt als de behuizing van de neurostimulator geleidend zijn.

Om af te sluiten, bieden we een kritische evaluatie van tegenmaatregelen die berusten op de fysiologische signalen van de patiënt voor het genereren van cryptografische sleutels tussen twee apparaten. Ons werk onthult ernstige beveiligingszwakheden in twee *pairing* protocollen die in de literatuur worden voorgesteld. Bovendien laten we zien dat de meeste van de bestaande tegenmaatregelen steunen op onrealistische veronderstellingen die de capaciteiten van de aanvaller onderschatten. Dit werk wordt afgesloten met een reeks aanbevelingen voor het veilig gebruik van fysiologische signalen in cryptografische protocollen.

Contents

List of Figures

List of Tables

Abbreviations

DAQ Data Acquisition System. 7

DoS Denial-of-Service. 13, 29, 36, 40

DPSK Differential Phase Shift Keying. 21

ECC Error Correction Code. 31

ECG ElectroCardioGram. 30, 37, 38

EMI ElectroMagnetic Interference. 18, 41, 46

FCC Federal Communications Commission. 13, 21

FDA U.S. Food and Drug Administration. 6, 14, 16, 18, 39

FSK Frequency Shift Keying. 21, 22

IC Integrated Circuit. 23

ICD Implantable Cardioverter Defibrillator. v, vi, 3, 9, 13, 14, 18, 25

IMD Implantable Medical Device. v, 3–9, 11–14, 17–21, 23, 25–33, 35–41, 43–46

JTAG Joint Test Action Group. 15, 23, 24

LFP Local Field Potential. 31

MAC Message Authentication Code. 27

MiTM Man-in-The-Middle. 31, 34, 38

OOB Out-Of-Band. 28

OOK On-Off Keying. 21, 22

PCB Printed Circuit Board. 23

PKI Public Key Infrastructure. 33, 37

RFID Radio-Frequency IDentification. 13, 28

SDR Software Defined Radio. 7, 12, 15, 16, 21

UART Universal Asynchronous Receiver/Transmitter. 15, 23, 24

Part I

Medical Device Security and Privacy

Chapter 1

Introduction

Implantable Medical Devices (IMDs) allow to monitor and control a broad range of diseases, such as cardiac arrhythmia, diabetes or Parkinson's disease. The most common IMDs include pacemakers, Implantable Cardioverter Defibrillators (ICDs), insulin pumps and neurostimulators. Already in 2001, the number of patients in the United States with an IMD exceeded 25 million [31]. Currently, seven to ten million people worldwide are living with Parkinson's disease [14], whereas diabetes affects more than 422 million people [23]. The number of people with chronic diseases that require an IMD will continue to rise because of people living an unhealthy lifestyle and the ageing of the population. For example, it is estimated that more than a million pacemakers will be implanted only in 2023, resulting in a revenue of more than 10 billion dollars [33].

Recent advancements in low-power wireless technologies and energy-efficient hardware platforms made it possible to develop IMDs with wireless capabilities which can deliver sophisticated treatments to patients. Once implanted in a patient, an IMD can only interact with the outside world through its **analogue interface** (i.e. internal sensors) and its **wireless interface** (i.e wireless bidirectional communication link), as shown in Figure. 1.1.

The internal sensors are used to constantly measure physiological signals in the patient's body, enabling the IMD to detect abnormal situations and actuate accordingly. For example, this allows an ICD to deliver an electrical shock to the patient's heart when this is not beating correctly[1].

[1]Some IMDs also have a built-in speaker that emits a sound when the IMD is not functioning correctly or before its battery is fully depleted.

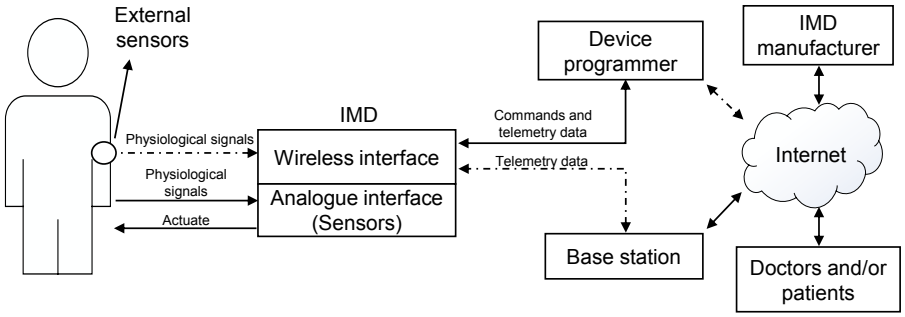

Figure 1.1: System architecture. The dashed lines denote links that do not always exist.

The wireless interface enables the IMD to communicate with several external devices such as device programmers, base stations and external sensors. Next, we focus on the wireless communication channel between the IMD and each of these peripherals.

By means of a device programmer, doctors cannot only gather telemetry data and patient's information stored in the IMD, but also modify the IMD's settings via the wireless interface. In some cases, device programmers have an Internet connection that allows IMD manufacturers to remotely update their software.

IMDs can often interact with base stations to monitor the patient's condition when he is at home. Base stations are placed in close proximity to the patient's bed in order to collect telemetry data from the IMD while the patient is sleeping. This telemetry data is then transmitted over phone or the Internet to a server that is often managed by the IMD's manufacturer. Doctors – and in some cases also patients – can log in to a website portal in order to review the telemetry data. Currently, there are several remote monitoring systems available in the market. In the near future, these remote monitoring systems could be extended to enable a bidirectional communication link between the base station and the IMD, allowing doctors in the hospital to remotely reprogram the patient's IMD when the patient is at home.

In addition to being able to interact with device programmers and base stations, IMDs can often also communicate with external sensors – either worn by the patient or implanted in his body – via the wireless interface. The data collected by the sensors is sent to the IMD so that the latter can dynamically adjust its

settings without requiring the doctor's intervention. An example of IMD-sensor communication would be an insulin pump system where the patient attaches a continuous glucose monitoring sensor to his abdomen that periodically and automatically measures and sends their glucose level to the insulin pump.

All these developments make IMDs more prone to failures and broaden their attack surface. As IMDs are devices that perform critical functions, they can jeopardise the patient's life if they stop working correctly due to unintentional or intentional reasons. According to Zian Tseng, a physician who investigated cardiac devices at the University of California, uninentional failures due to IMD malfunctions occur more often than expected [34]. Surprisingly, half of the cardiac devices he analysed had serious defects ranging from drained batteries to mechanical failures. Similarly, intentional failures in IMDs due to attacks by adversaries pose an important threat. Adversaries could control the patient's IMD by sending maliciously crafted commands to alter its settings or turn off the therapies. Furthermore, adversaries could compromise the patient's privacy by intercepting the messages exchanged over the air between the device programmer and the IMD. These messages contain highly sensitive information including personal patient data, such as their address and social security number, and medical data such as information about the treatment and telemetry.

There are several reasons why someone may want to launch attacks against patients with an IMD.

- **Cause physical harm.** Criminal organizations could threaten and blackmail patients, or even worse, hurt or kill them. Although there are no known real attacks against patients with an IMD, in 2007, the former Vice-President of the United States, Dick Cheney, asked his doctor to disable the wireless interface of his pacemaker to thwart possible assassination attempts [20].
- **Economical.** Individuals, companies or potential competitors of IMD manufacturers could influence stock markets to make profit. For example, in 2016, a team of security researchers called MedSec allegedly investigated the security of several St. Jude Medical implanted heart devices, and concluded that these devices had serious security vulnerabilities [28–30]. Rather than notifying their findings to St. Jude Medical, MedSec released a report jointly with the investment firm Muddy Waters, causing St. Jude shares to fall 10 percent. Muddy Waters and MedSec admitted that they had a financial agreement to profit from a fall in St. Jude's stock price.
- **Privacy breaches.** IMDs collect highly privacy-sensitive data from the patient's body. For example, by analysing the data gathered by a pacemaker, one can learn the patient's physical activity pattern, making

it even possible to infer when the patient had sex. In this thesis, we focus on how adversaries could get access to this data in an illegitimate way. However, even if unauthorised access to this data is prevented, some privacy risks remain. For example, there is a risk that this data is used for other purposes than the initially intended one. The latter case is out of the scope of this thesis.

- **Tracking.** The messages exchanged over the air between the device programmer and the IMD typically include information that can uniquely identify patients such as the unique IMD's serial number. Thus, adversaries could track, locate or identify patients simply by intercepting the messages sent by IMDs. If adversaries install beacons in strategic places throughout a city, they could even learn the patients' movement patterns.

1.1 Contributions of the Thesis

The main contribution of this thesis is to advance the understanding of security and privacy in IMDs. We mainly contributed to this field in two ways: (i) by analysing the security and discovering vulnerabilities in several widely used IMDs and (ii) by proposing practical and effective countermeasures to address the security weaknesses we have identified. The mapping of the contributions to our papers is shown in Table 1.1.

Our work on identifying vulnerabilities in IMDs had considerable real-world impact and received significant media coverage in newspapers, radio and TV shows. We followed the principles of responsible disclosure and notified the affected manufacturers six months before publishing our results. In all the papers, we deliberately omitted some information so that our work does not help someone with malicious intentions to perform attacks on real patients. We had discussions with the affected manufacturers to evaluate the risks of these attacks and suggested possible mitigation strategies. All manufacturers acknowledged our findings and some of them are currently considering adding security mechanisms in their devices as a result of our work. In addition, recent research from our team and other researchers has led the U.S. Food and Drug Administration (FDA) to release several documents with recomendations for mitigating and managing cyber-security threats in IMDs [22]. Currently, the FDA is responsible only for ensuring that IMDs are sufficiently safe and effective to be implanted within the patient's body. However, in the near future the FDA will additionally check if IMDs comply with their security guidelines before giving its approval.

In detail, our contributions can be summarised as follows:

C1) Security analysis of IMDs. We analysed the security of the wireless protocol between a device programmer and widely used IMDs. This includes Implantable Cardioverter Defibrillators (ICDs), insulin pumps and neurostimulators from the leading IMD manufacturers. As IMDs typically utilise proprietary protocols for which no information is available, we fully reverse engineered the protocols using a non-invasive black-box approach and inexpensive hardware devices. The reverse engineering led us to document the protocols in use and to find the protocol state-machine. Our main goal was to demonstrate that proprietary protocols can be reverse engineered by adversaries with limited resources who do not have physical access to the devices but can only eavesdrop on the wireless communications. Our work has resulted in the identification of important security weaknesses in IMDs used by millions of patients worldwide.

C2) Wireless attacks on IMDs. We demonstrated passive and active wireless attacks against various types of IMDs which could jeopardise the patient's life or compromise the patient's privacy. To mount these attacks, we used commercial antennas along with inexpensive hardware equipment such as Software Defined Radios (SDRs) and Data Acquisition Systems (DAQs). Moreover, we investigated the feasibility of these attacks, and showed how to overcome some of the challenges that adversaries would face to carry out these attacks in practice.

C3) Study the feasibility of using cryptography in IMDs. We conducted the first study on whether cryptography can be used in IMDs without significantly reducing their battery lifetimes. Our work targeted a wearable insulin pump that wirelessly communicates with a remote control. We presented an AES-based solution that is optimised for energy consumption, and implemented it in a micro-controller similar to the one used in the insulin pump. The choice of AES was based on the hypothesis that the computation cost would be negligible compared to the communication cost. We proposed several variants of our solution and measured for each of them the extra computation and communication energy cost due to the use of cryptography in both the remote control and the insulin pump. Finally, we elaborated on possible ways to optimise the message format in order to decrease the total energy cost.

C4) Novel key establishment solutions for IMDs. We proposed novel cryptographic solutions for a device programmer and an IMD to securely establish a session key. We consider the case in which the devices share a long-term key beforehand, as well as the case in which the devices do not share any prior secrets. For the former case, we designed a novel semi-offline key agreement protocol where the master key is updated periodically. This limits the

impact of attacks that aim at compromising the master key stored in all device programmers. Each time a new master key is generated, the IMD can update its device-specific key using bilinear pairings. For the latter case, we proposed a low-cost source of randomness for IMD as well as a novel technique for securely and reliably transporting a key from the IMD to the device programmer.

C5) Security evaluation and framework of countermeasures based on physiological signals. We conducted a critical evaluation of countermeasures that allow a device programmer and an IMD to generate and establish a key through the use of a patient's physiological signal. To this end, we first challenged several assumptions made by manufacturers and demonstrated that some of them do not hold in practical scenarios. Our work revealed serious security weaknesses in two pairing protocols proposed in the literature, namely the H2H and Biosec protocols. This shows that it is a non-trivial task to design secure and lightweight pairing protocols for IMDs based on the use of a patient's physiological signal. Motivated by our findings, we provided a set of security design guidelines in which we specified the requirements physiological signals should satisfy to be used in cryptographic protocols. Furthermore, we described how to use patient's physiological signals in combination with fuzzy extractors securely while keeping the energy cost in the IMD as low as possible.

C6) Design of a privacy preserving system for remote healthcare monitoring and reprogramming. We presented a protocol by which an IMD can establish an end-to-end secure channel with a hospital. This allows remote monitoring and reprogramming of an IMD when the patient is at home. However, even if cryptography is used, the context information of these communications, also known as metadata, could leak sensitive information about patients. For example, unauthorised entities or adversaries could infer the patients' health condition or discover which disease they have. Our protocol has been designed to conceal this information and hence it provides privacy to patients. As a result, unauthorised entities are unable to learn to whom the medical data belongs or to which hospital the data is sent. To achieve this, we utilised cryptography and an anonymous communication channel.

Table 1.1: Mapping of contributions to our papers.

Paper/Contribution	C1	C2	C3	C4	C5	C6
A Privacy-preserving Remote Healthcare System Offering End-to-End Security						✓
On the Feasibility of Cryptography for a Wireless Insulin Pump System	✓	✓	✓			
On the (in)security of the Latest Generation Implantable Cardiac Defibrillators and How to Secure Them	✓	✓		✓		
Securing Wireless Neurostimulators	✓	✓		✓		
A Critical Evaluation of Security Solutions based on Physiological Signals					✓	

1.2 Structure of the Thesis

This thesis mainly comprises of two parts. In Part I, we define the objectives, contributions and motivation behind this thesis. In Chapter 2 we give an overview of the proposed attacks against IMDs to place our work into context. Chapter 3 describes several methodologies and techniques for reverse engineering proprietary protocols. In Chapter 4 we group and summarise the existing countermeasures including those that we have proposed. Finally, Part I gives concluding remarks and presents potential research directions for future work.

Part II consists of five publications:

1. We present a privacy-preserving protocol for establishing an end-to-end secure channel between an IMD and a hospital. This protocol can be used for remote monitoring and reprogramming of the patient's IMD when the patient is at home [90].

2. We analyse the security of an insulin pump system, and study the feasibility of using cryptography to create a secure wireless channel between the insulin pump and the remote control [92].

3. We evaluate the security of some of the latest generation ICDs, and propose several short and long-term countermeasures [91].

4. We analyse the security of a neurostimulator, and present a security architecture for securing the wireless channel between the neurostimulator and the device programmer [93].

5. We provide a critical evaluation of security solutions based on the use of patient's physiological signals for generating cryptographic keys. Furthermore, we provide a set of recommendations on how to securely use patient's physiological signals in cryptographic protocols.

Chapter 2

Attacks on Medical Devices

Prior work has investigated possible ways of conducting attacks against IMDs. We group the proposed attacks into two main categories depending on the type of interface that adversaries need to exploit them (see Figure. 1.1). This includes (i) wireless attacks where adversaries leverage the wireless nature of the communication to perform passive and active attacks and (ii) analogue attacks whereby adversaries tamper with the readings taken by the sensors embedded on IMDs to cause the latter to actuate incorrectly.

While analogue attacks on IMDs have received little research attention until now, there has been a significant amount of research on studying wireless attacks on IMDs over the last decade. The main challenge that this research had to face is the lack of publicly available information about the specifications of the wireless protocols being used. This is because IMD's manufacturers typically use proprietary (i.e. non-standard) protocols whose specifications are kept secret as a means to provide security, also known as security-through-obscurity.

2.1 Wireless Attacks

This section first reviews several assumptions that are widely adopted by IMD's manufacturers which do not hold in practice. Subsequently, we subdivide the proposed wireless attacks according to device type. This includes (i) *attacks on implantable cardiac devices*, (ii) *attacks on insulin pump and infusion systems* and (iii) *attacks on neurostimulators*. Note that there exist other IMDs with wireless capabilities but their security has not yet been analysed.

2.1.1 Are Wireless Attacks Realistic?

Manufacturers often underestimate the adversaries' capabilities. Therefore, they make unrealistic security assumptions which can significantly affect the security of IMDs.

- **Need for sophisticated hardware equipment**: Before developing a security solution, an adversarial model should be defined that is as realistic as possible. This is important because the security mechanisms are designed to protect only against the type of adversaries specified in the adversarial model. Researchers typically consider adversaries who can eavesdrop the wireless channel, modify and inject messages or have physical access to the devices. Despite this type of adversary is widely accepted within the security community, manufacturers often consider weaker adversaries with limited resources and capabilities.

  It is important to stress that, as technology advances, the price of the necessary hardware equipment decreases. This enables adversaries to execute complex attacks using only commercial and inexpensive devices. For example, there is a 32 dollar device available in the market for conducting eavesdropping-and-jamming attacks to compromise the security of rolling-code-based systems [25]. These systems are currently used in most cars and garage doors. Other examples include SDR-based devices such as the HackRF One [61], bladeRF [99], YARD Stick One [62] and USRP [12]. All these devices can be purchased online for a few hundred dollars.

  Another assumption that does not hold in practice is that adversaries cannot conduct their attacks without getting noticed, due to the size of their hardware devices and their antennas. There are already examples that show how hardware devices can be miniaturised to fit in a backpack [78]. Furthermore, there is a substantial number of research papers that focus on designing inexpensive and small antennas with reasonable gain and directivity. These antennas could even be attached to the adversaries' clothes, enabling adversaries to hide these antennas in the pockets of a jacket or under a sweater.

- **Physical range constraints**: Most IMDs with wireless capabilities can communicate with device programmers over a few meters. A common assumption is that adversaries need to be a few meters away from the targeted device to be able to capture or transmit messages. Nevertheless, adversaries can use sophisticated hardware equipment along with directional antennas to extend the distance from which they can intercept these messages. Furthermore, adversaries do not need to follow

the Federal Communications Commission (FCC) regulations and can transmit their messages using a higher power. This could enable them to extend the communication range even up to a few orders of magnitude. For example, BlueSniper enables to mount attacks on Bluetooth devices from more than 1.5 km [50]. Similarly, a long range Radio-Frequency IDentification (RFID) reader can be modified to clone a RFID tag from up to 1 m away [60]. Extending such short-range communication channels even further is however unrealistic due to the significant increase in antenna size.

2.1.2 Attacks on Implantable Cardiac Devices

The first security analysis of an ICD with wireless capabilities was conducted by Halperin et al. in 2008 [70]. They partially reverse-engineered the proprietary protocol between a device programmer and an ICD over short-range communication (less than 10 cm). For the reverse engineering, they followed a black-box methodology (see Section 3 for more details about this approach). Their study revealed that the transmissions sent over the air between the device programmer and the ICD are neither encrypted nor authenticated. This led them to perform several types of attacks on ICDs simply by replaying messages previously transmitted by legitimate device programmers. Although these attacks could compromise the safety and the privacy of patients, their impact is quite limited. Firstly, adversaries would need to be in close proximity (few centimetres) with the patient, not only to intercept the messages sent by these devices while there is an ongoing communication, but also to replay the captured messages to the ICD at a later stage. Furthermore, adversaries would be able to replay only those messages that were previously sent by device programmers.

Hei et al. introduced various Denial-of-Service (DoS) wireless attacks that can be performed against any type of IMD without needing to reverse engineer the proprietary protocol [73]. Their goal is to reduce the IMD's battery lifetime from several years to only a few weeks. These attacks leverage the fact that IMDs need to perform several computations when receiving a new message in order to check if it is valid. By repeatedly sending messages, adversaries can force the IMD to verify if their messages are valid, which could lead to a significant overhead in terms of energy consumption. Unfortunately, these attacks are difficult to defeat and cannot be prevented with traditional cryptographic solutions.

A few years later, Barnaby Jack demonstrated several attacks on pacemakers at the BreakPoint security conference in Melbourne. However, the exact details of his work are not known [19].

Our contribution. In [91], we analysed the security of the latest generation of ICDs from one of the leading IMD manufacturers. In contrast to the work by Halperin et al. [70], our work focused on the protocol used between the device programmer and the ICD over a long-range communication (few meters). As the protocol was proprietary, the first step of our analysis was to fully reverse engineer it. By using a black-box methodology, we studied the feasibility of reverse engineering the proprietary protocol by a weak adversary who does not have physical access to the devices but can only eavesdrop on wireless communications. We demonstrated that the proprietary protocol has serious design and implementation weaknesses. Our work revealed the first known attempt by a IMD manufacturer to obfuscate the data that is transmitted over the air. The data is obfuscated by XORing it with a bit sequence generated from a Linear Feedback Shift Register (LFSR). This LFSR sequence is constant throughout sessions, and is reused in all ICDs we analysed. We validated that our findings apply at least to 10 different types of ICDs that are currently on the market.

After learning the inner-workings of the protocol, we were able to carry out attacks that could compromise the ICD's availability and violate the patient's privacy. Moreover, we demonstrated that replay and spoofing attacks are also possible. As the ICDs we studied are activated by the device programmer over short-range communication (few centimetres), these attacks could suffer from the same limitations as the attacks proposed by Halperin et al. [70]. However, we found that ICDs remain in a semi-active mode for 5 min every time a communication session with a device programmer is terminated. While being in this mode, ICDs accept messages sent over the long-range communication channel. Adversaries could thus exploit this 5 minute time window to mount attacks from several meters away. The only limitation for adversaries is that they need to be present at the specific time that the patient has his ICD in semi-active mode.

Follow-up work. As previously explained, MedSec investigated the security of several implantable cardiac devices from St. Jude Medical. Subsequently, MedSec released a report where they claimed to have found serious security vulnerabilities in these devices [29]. St. Jude Medical denied the veracity of this report and claimed that MedSec analysed an old software version that was no longer used on their device programmers. However, the FDA has recently issued a (voluntary) recall to patch security weaknesses in 465,000 St. Jude pacemakers [32]. Unfortunately, updating the firmware of these devices comes with risks associated to possible device failures. For example, St. Jude stated that the devices can stop working after the update in 0.003% of cases.

Recently, Rios and Butts have identified serious security weaknesses in implantable cardiac devices from the four leading manufacturers [112]. Instead

of using a black-box methodology similar to ours, they studied the feasibility of obtaining the firmware from a device programmer or a base station and analysing it. Thus, they implicitly considered an adversary who possesses or has temporary access to one of these devices. Their findings revealed that the Joint Test Action Group (JTAG) and Universal Asynchronous Receiver/Transmitter (UART) debugging interfaces are not disabled, making it possible for adversaries to obtain the firmware of these devices. Since no firmware packing, obfuscation or encryption techniques are used, the firmware could be reverse engineered easily (see Section 3 for more details about this approach). This task is made even easier due to the use of ASCII text in function names, source code comments and software debugging symbols. Moreover, they found sensitive information, such as hard-coded credentials and patient's data, stored in the clear in device programmers and base stations. The patient's data included physicians' names, phone numbers, social security numbers or treatment data. Additionally, they reported that both device programmers and base stations lack security mechanisms to validate the source of the entity that distributes the firmware for these devices. Adversaries can thus exploit this weakness to upload malicious firmware in these devices. Finally, they discovered that the device programmer's firmware include third-party libraries that contain many vulnerabilities. For example, one vendor has more than 3500 vulnerabilities in its third-party components.

2.1.3 Attacks on Insulin Pumps Systems and Infusion Systems

In 2011, Jay Radcliffe, a diabetic patient, analysed the security of his insulin pump and continuous glucose monitoring system [106]. He captured several messages sent by these devices using an Arduino board, and then reverse-engineered parts of the proprietary protocol. For this purpose, he did not only analyse the captured messages but also inspected the Java library file (i.e. JAR file) included in the insulin pump. The latter was possible because these files were not obfuscated, thus facilitating the analysis.

In the same year, Li et al. reverse engineered the proprietary wireless communication protocol between an insulin pump and a remote control [86]. To show what adversaries could do after learning how the protocol works, they conducted several passive and active attacks against an insulin pump from 20 meters away. The authors stated that it would be possible to extend this range even further by using an antenna with more directivity or a SDR that allows to transmit messages with higher power.

In 2015, Billy Rios found important security weaknesses in the Hospira's symbiq, an infusion system that can be used to deliver medications by pumping them

directly into the patient's bloodstream [26]. This system has a wired or wireless connection to communicate with the hospital's information system (HIS). His findings indicated that adversaries could fully control the infusion pump to change the dosage that is delivered to the patient. This could lead to over- or under-infusion of critical patient therapies. The FDA acknowledged these findings and recommended hospitals to stop using this infusion system [27]. Hospira stated that they will develop a software update to patch these problems. This patch is intended to close some ports of the infusion system that were initially open and to provide extra security protection.

In 2016, Jay Radcliffe analysed the security of the Animas OneTouch Ping insulin pump system [107]. This work reported that this system does not have any security mechanisms to protect the wireless data exchange between the devices.

Our contribution. In [92], we evaluated the security and privacy properties of the wireless communication between a widely used insulin pump and all its peripherals. This includes a remote control, a glucose meter, a continuous glucose monitor system and a USB stick. To achieve our goal, we first fully reverse engineered the proprietary protocol between the insulin pump and its peripherals using a black-box methodology. Prior to reverse engineering the protocol, we had to discover several wireless communication parameters used by these devices, such as the transmission frequency, modulation scheme and symbol rate. Subsequently, we created our own receiver and configured it with these wireless communication parameters to capture the messages sent by these devices. Our reverse engineering of the protocol resulted in the identification of the message format and the protocol state-machine in use. Our work demonstrated that all messages transmitted between these devices are sent both unencrypted and unauthenticated. We then went one step further and showed that this knowledge can be used to mount several passive and active attacks on the insulin pump.

Follow-up work. Reverberi and Oswald investigated the security of a continuous glucose monitor system with wireless capabilities called Dexcom G4 [111]. This system consists of a sensor (implanted) that is connected to a transmitter, and a hand-held receiver device that collects the messages sent by the transmitter and displays the patient's glucose level. They started their analysis by removing the plastic packaging covering the transmitter to find the exact micro-controller and transceiver being used. Subsequently, they identified the debug pins of the micro-controller and connected a suitable programmer to them for extracting the device's firmware. Yet, they were unable to recover the device's firmware because the read-out protection bit of the micro-controller was set. At this stage, they opted for capturing messages sent by the transmitter using a SDR. After analysing these messages, they came to

similar findings compared to ours; the Dexcom G4 protocol does not employ any security mechanisms and the transmitted messages are neither encrypted nor authenticated. Finally, they devised a series of attacks against this system. This ranged from spoofing attacks that aim at maliciously changing the sensor's readings to privacy attacks whose goal is to track patients based on the messages sent by their transmitters.

Skorobogatov detailed the main challenges for extracting the firmware from an IMD, and discussed how to overcome them [125]. His work targeted a wireless tubeless insulin pump manufactured by Insulet called OmniPod. By using a combination of hardware techniques, he was able to find the circuit diagram of the device and identify the locations of the reset and debug pins in the chip. Furthermore, he described possible ways of disassembling the firmware once it has been extracted. The analysis of the disassembled code to find the inner workings of the wireless protocol was left as a future work.

2.1.4 Attacks on Neurostimulators

Our contribution. In [93], we performed the first security analysis of the wireless protocol between a device programmer and a neurostimulator over short-range communication (less than 10 cm). We focused on a commercial neurostimulator from one of the leading manufacturers. Due to the lack of publicly available information on how the protocol works, we first had to fully reverse engineer the protocol to determine the message format and the protocol state-machine. This was achieved following a black-box reverse engineering methodology and using inexpensive hardware devices. Our evaluation revealed that the wireless protocol does not use any cryptographic mechanisms. While these security weaknesses could be exploited to execute attacks against neurostimulators, these attacks suffer from several limitations in practice. Firstly, adversaries would need to know the serial number (SN) of the neurostimulator. Intuitively, one possible way for adversaries to get the neurostimulator's SN would be to eavesdrop the wireless channel to capture the messages exchanged between these devices while there is an ongoing session. However, we found that neurostimulators have an important implementation flaw that makes them accept messages with an empty neurostimulator SN field. This means that adversaries could create a valid message, without a SN, and reuse it for all neurostimulators. Secondly, another limitation is that adversaries would need to be in close proximity to the patient to be able to communicate with his neurostimulator. We argue that there are definitely several scenarios, such as a crowded subway, where this would be possible, but this clearly reduces the impact of our attacks.

2.2 Analogue Attacks

All electronic circuits have components that can behave as an antenna under certain circumstances. When this occurs, these components can capture ElectroMagnetic Interference (EMI) radiation produced by other devices such as Magnetic Resonance Imaging (MRI) equipment. The FDA has acknowledged that EMI could lead IMDs to malfunction and has taken measures to reduce the risks. Making IMDs resistant to *unintentional, high-power* EMI has been the focus of a substantial amount of research over the last years. Yet, several incidents have been reported allegedly related to EMI in IMDs [8].

Recently, Foo Kune et al. investigated the feasibility of injecting *intentional* EMI signals on the sensors embedded on ICDs to maliciously alter their readings [84]. The goal of these attacks is to convince the ICD that the patient's heart is not beating correctly when it actually does so. To execute these attacks, one could inject high-power EMI signals. However, high-power EMI signals are difficult to generate and could disable the electronic components of the equipment used by adversaries. Instead, the authors proposed to inject *intentional, low-power EMI* signals since this can cause fluctuations on the order of millivolts which are sufficient to influence the sensor's readings.

They proposed two attacks: (i) *baseband EMI attacks* and (ii) *amplitude-modulated EMI attacks*. The former relies on injecting signals within the sensor's frequency band (i.e. baseband) whereas the latter is based on injecting signals on a frequency outside the baseband but still within the frequency band where the sensor responds. As amplitude-modulated EMI attacks target devices that lack filters and IMDs have filters to remove undesired signals outside the baseband, these attacks do not pose a real threat for IMDs. To conduct baseband EMI attacks, the first step is to find the *resonant frequency* of the sensor. They performed a series of experiments where they swept through all possible IMD's frequencies (between 0.1-1 KHz), and observed at which frequency band the signals had the strongest amplitude. These attacks could cause pacing inhibition or induce defibrillation from 1-2 meters. However, the authors stated that, when testing these attacks in a more realistic setup, the range from which these attacks can be conducted decreases to 5 cm. It is unclear whether these attacks could be executed from further away if a more powerful EMI source is used.

Chapter 3

Reverse-Engineering Methodologies

Most IMDs use proprietary (i.e. non-standard) protocols to wirelessly communicate with device programmers. IMD manufacturers rely on maintaining the protocol specifications secret as a means to provide "security", i.e. security-through-obscurity. They assume that adversaries who do not have access to the protocol specifications, will not be capable of learning its inner-workings. However, several papers have already shown that proprietary protocols can be reverse engineered without prior knowledge.

Severe weaknesses were found in proprietary protocols used in other application domains. For example, Rouf et al. analysed the security and privacy of Automatic Meter Readings (AMR) systems. These systems allow a receiver device when being in close vicinity to a house to wirelessly gather electricity, gas and water consumption data [115]. By reverse engineering the protocol being used, they determined that these systems lack security mechanisms to protect the confidentiality and integrity of the consumption data. Similarly, Smith et al. reverse engineered the wireless proprietary protocol between airplanes and ground stations, which is also known as the Aircraft Communications Addressing and Reporting System (ACARS) [126]. Their work revealed that 99% of the traffic is unencrypted and that it is possible to recover the key used in the substitution cipher in order to decrypt the 1% of encrypted traffic.

3.1 Overview

Intuitively, the easiest way for someone to learn how a protocol works would be to steal its specifications through information leaks, social engineering attacks or intrusion into computer systems. However, these attacks are not always possible. Protocol reverse engineering implies finding both the message format and how messages are exchanged between two devices (i.e. protocol state-machine) without knowing its specifications. There are several techniques for reverse engineering a protocol, each of which has its own advantages and disadvantages. This includes (i) *black-box reverse engineering* and (ii) *firmware reverse engineering*. The former is non-invasive and does not require to have physical access to the device, whereas the latter is often invasive and requires either to have physical access to the device itself or to get a copy of it. In practice, the choice of which methodology to use will mainly depend on the information available about the device and if it can be broken. Next, we describe each of these reverse engineering techniques more in detail.

3.2 Black-box Reverse Engineering

Black-box reverse engineering is a technique that allows to infer the inner-workings of a protocol simply by providing inputs to a device and observing its outputs. In this case, one can use a device programmer to perform an operation on an IMD and then inspect the format of the messages transmitted as a result of this operation. We acknowledge that having access to a device programmer to choose which operations are performed can speed up the process of reverse engineering. Even if the process of reverse engineering may take longer, adversaries can still learn the protocol inner-workings when they do not know which actions are performed on the device programmer.

Through a device programmer one could, for example, modify the patient's name, stored on the IMD, and intercept the messages exchanged between the device programmer and the IMD. By slightly changing the input each time and performing differential analysis, it is possible to learn where the bits corresponding to the patient's name are placed within the message and how this data is encoded. By following this methodology and conducting a thorough analysis of the captured messages, one can infer all the message fields.

To reverse engineer a protocol using a black-box methodology, several messages sent by the devices need to be first captured while performing different operations on the IMD, e.g. changing the patient's name. To this end, one can either fabricate a receiver device, manipulate an existing receiver device or use an

SDR and then connect an antenna to it. As antennas operate in a special range of frequency, it is important to design or buy an antenna that can intercept the signals sent by the device programmer and the IMD. The antenna's gain will be one of the factors that will determine from which distance the messages can be captured. However, the larger the antenna gain, the more expensive the antenna will be.

The process of reverse engineering a protocol using a black-box methodology can be divided into three main steps. Next, we describe each of these steps more in detail.

1) Get public information from the Internet: All wireless devices sold in the U.S. have a FCC ID that contains all the information about their internal functioning such as circuit diagrams, block diagrams or internal photographs [5]. However, IMD manufacturers typically argue that this information should remain confidential. According to a confidentiality request by Medtronic, one of the world's leading IMD manufacturers, *"disclosure would, in effect, give away the fruits of the labours of Medtronic's engineering personnel, who have designed the equipment and the manufacturing processes. Disclosure would also offer competitors additional unwarranted insight into the state of product development thereby allowing such competitors an advantage that would not be available to Medtronic"* [18].

2) Find the wireless communication parameters: As most information about these devices is kept secret, an important step is to identify the wireless communication parameters used by the devices.

- **Transmission frequency**: The transmission frequency of these devices is typically one of the few data that can be obtained through its FCC ID. Both device programmers and IMDs have their own FCC ID, which is typically printed on the back of the device.
- **Modulation scheme**: The modulation scheme can be found by intercepting messages and analysing the waveform of the signals both in the time and frequency domain [105]. Due to the asymmetry between device programmers and IMDs, it is not uncommon to find two distinct modulation schemes depending on the direction of the communication. Examples of modulation schemes used in device programmer - IMD communications are Frequency Shift Keying (FSK), Differential Phase Shift Keying (DPSK) or On-Off Keying (OOK).
- **Symbol rate**: Discovering the duration of the modulated bits, also known as symbols, just by examining the signal's waveform is often a very difficult task. This mainly depends on the type of modulation being used. For

example, when using OOK, the '1s' and '0s' can be easily distinguished allowing to measure the symbol rate. In some other cases, it is possible to look at the raw bits (i.e. bits before modulation) to be transmitted by opening the device and tapping one pin of its micro-controller [70, 92]. This could be seen as a *semi black-box approach*. However, sometimes neither of the previous two approaches can be used to find the symbol rate. For those cases, we developed a technique for estimating the symbol rate based on the Hilbert transform [91].

Our technique requires (at least) one message or parts of it whose bits are known. For this purpose, one can leverage the synchronization sequence at the beginning of each message, which typically contains a series of alternating '1s' and '0s' sent consecutively. Our technique can be applied to any modulation scheme in which the information is embedded in the frequency or the phase of the signal (e.g. FSK).

However, this technique does not provide the exact symbol rate value, but gives a small range of possible symbol rate values. To find the exact symbol rate, we created a program that performs a sweep over all possible symbol rate values within this range, increasing the symbol rate by one symbol each time. For each iteration, our program demodulates several messages, and checks whether the demodulated bits are equal for all the messages. This allows to find the symbol rate being used by the devices, as the correct symbol rate is the one for which no bit errors are produced.

3) Capture and analyse messages sent by the devices: Subsequently, one can create a receiver program for capturing and demodulating the messages sent by the devices. We observed that if the receiver program uses slightly different wireless communication parameters compared to those used by the devices, this leads to an incorrect demodulation of the messages, i.e. messages with erroneous bits. After demodulating the messages, one can distinguish the bits containing the information from the noise by looking at specific bit sequences that indicate the beginning and the end of the packet. Next, the packets can be thoroughly analysed to identify common patterns and to find dependencies among message fields. As protocols often use the same or similar message fields, knowledge about other wireless protocols can help to determine which message fields are used.

3.3 Firmware Reverse Engineering

In addition to reverse engineering a proprietary protocol through a black-box approach, the protocol can be reverse engineered by retrieving and analysing

the firmware of a device programmer, a base station or an IMD. Ideally, these devices should implement security mechanisms to prevent anyone from retrieving and analysing their firmware. For example, the device's debugging interfaces could be disabled to make it more difficult to retrieve the device's firmware. Both obfuscation and encryption could also be used to ensure that the device's firmware is unreadable to anyone who manages to retrieve the firmware and decompile or disassemble it. In practice, though, most device programmers, base stations and IMDs do not employ any of these security mechanisms.

Next, we describe the process of how to extract the firmware from one of these devices and analyse it to learn how the proprietary protocol works.

- **Component identification and analysis:** The first step is to open the device in order to explore the Integrated Circuit (IC) on its Printed Circuit Board (PCB). In some cases, it is possible to identify the chip and obtain its data sheet on the Internet. This makes it easier to find the debugging and programming pins. However, many devices use chips for which no public information is available, which significantly increases the difficulty of retrieving the firmware. In such case, X-Ray imaging can help to discover the internal layers of the PCB and its connections. The main limitation of this technique is that it requires specialised hardware equipment. A more inexpensive approach to find the interconnections between components on the PCB is to de-solder all its electronic components, clean the PCB with a solvent and dry it. This process results in a PCB where all wires are easily traceable. Yet, this approach is only suitable for PCBs with one or two layers.

  To discover the functions of the micro-controller's pins, one can either use a logic analyser or measure the voltage on the micro-controller's pins during the device operations. Both approaches allow to reduce the number of pins that are suitable for the debugging and programming interfaces.

- **Firmware extraction:** The easiest way to retrieve the device's firmware is to use any of the programming or debugging interfaces of the chip such as JTAG and UART. These interfaces can be used not only to get the device's firmware but also to trace instructions or to read memory sections. Thus, manufacturers should disable these interfaces in all production devices. However, Rios and Butts found that the JTAG and UART debugging interfaces of device programmers and base stations are not disabled [112]. In this case, any universal programmer can be used to get the firmware from these devices.

  If the read-out protection bit is set, one can attempt to reverse this operation in several ways. For example, one could perform voltage glitching

attacks to introduce a fault during power up of the device in order to circumvent the read-out protection bit [97]. Another way to bypass the micro-controller's read-out protection mechanism would be to use the technique proposed by Garcia et al. [63]. This technique can be applied to micro-controllers whose memory is divided into a number of blocks, where each block has its own access control bits that determine if the block is readable and/or writeable. The core idea behind this technique is to carefully erase part of the chip's memory in order to reset the access control bits.

This technique requires to have two copies of the chip, and works as follows. The first step is to erase block 0 on one of the chips in order to reset its access control bits to the default settings, i.e. readable and writeable. Subsequently, a small program can be written on block 0 that reads blocks 1,...,n and then outputs the data obtained in each of the blocks via one of the micro-controller's output pins. This procedure allows to recover blocks 1,...,n. For recovering block 0, one can proceed in a similar way and first erase blocks $1, ..., n$ in the other chip and then write a small program in block n to read block 0.

- **Code analysis:** As previously explained, most chips contain debugging or programming interfaces, such as JTAG or UART. The use of these interfaces makes it possible to observe the current state of the internal memory and registers, and set breakpoints either on a specific value of the program counter or on a certain memory or register condition. To disassemble the firmware and debug its code, one can use well-known tools such as IDA Pro [58], Binary Ninja [2] and MPLAB [9]. In addition to performing dynamic analysis, static analysis can help to reverse engineer parts of the protocol.

Chapter 4

Security Solutions

This section focuses on giving an overview of the countermeasures that have been proposed for addressing the wireless attacks and analogue attacks on IMDs. Prior to this, we describe what makes security of IMDs unique and why it is not straightforward to implement security mechanisms in these devices.

4.1 Problem Statement

While security and privacy are two important requirements that should be considered already during the design phase, adding security mechanisms to IMDs is a non-trivial task.

IMDs are small resource-constrained devices with a limited computation power and memory storage. They typically contain a custom ultra-low power microprocessor with 128 Kbytes of RAM. Most IMDs are operated by a single non-rechargeable and non-replaceable battery with a capacity between 0.5 A·h to 2 A·h [116][1]. This means that IMDs are required to have a low peak power and a low duty cycle. The IMD's battery typically lasts between five and seven years depending on the type of device and treatment. For example, the pacing pulses delivered by ICDs require about 25 µJ whereas a single electric shock can consume up to 40 J [69]. Once the battery is drained, the IMD is replaced through a surgical intervention which introduces a small, but non negligible risk of infections, sometimes even with lethal consequences. Furthermore, IMDs lack

[1]Some IMDs contain a rechargeable battery. However, this is not very common due to practical, economic and safety reasons.

input and output interfaces and cannot be physically accessed once implanted. Finally, several inherent tensions exist between some of the IMD's functional requirements and the required security and privacy goals. Below, we describe each of these tensions more in detail.

1) Security vs. safety: Implementing security mechanisms in IMDs inevitably increases the number of code lines and consequently the IMD's complexity. This makes IMDs more prone to software bugs. There are several well-known formal techniques and tools to detect bugs but experience has shown that it is difficult to have a bug-free implementation. For example, a software bug was found in a radiation therapy machine called the Therac-25 which caused serious injury and the death of several patients [85]. Because of this software bug, patients received radiation doses that were hundreds of times greater than the intended therapy. In this domain, it is paramount for IMDs to have a means to revert to a safe default mode if a bug is found during normal operation. In addition to the possibility of introducing software bugs, adding security could also reduce the IMD's reliability. For example, suppose that the device programmer and the IMD produce a different cryptographic key after executing a key establishment protocol. In such a situation, it is impossible for IMDs to distinguish between an error in the key establishment protocol and an adversary who attempts to send malicious messages.

2) Security vs. permissive access control policies: IMDs should implement security mechanisms to prevent security attacks but at the same time they should also develop permissive access control policies to grant medical staff access to the patient's IMD in emergencies. In non-emergency situations, patients typically go to a small set of doctors and there are no strict time requirements. Doctors can easily authenticate themselves to a server that contains all the cryptographic keys in order to retrieve the key for accessing the patient's IMD. However, this situation becomes more complicated in an emergency situation. Suppose an scenario where a patient with an IMD has an accident while he is abroad. Medical staff should be able to detect the IMD's presence and know what type of IMD the patient has, gather data stored in the IMD or reprogram it[2]. To further complicate matters, medical staff do not know the patient and could be unable to identify him; he may be unconscious or may not carry his ID card. In emergencies, a small delay when contacting the IMD's manufacturer or the patient's doctor to obtain the key could prevent the patient from receiving care on time.

This problem could be mitigated by defining various security levels depending on the type of action that doctors want to conduct. It is clear that having access

[2]Detecting the presence of an IMD is very important since some IMDs need to be deactivated before a surgery.

to the data stored in the IMD does not pose the same risks as having the ability to reprogram the IMD. Nevertheless, this may not be a viable solution since doctors should be able to perform all actions on the patient's IMD. This shows that there is need for a balance between security in non-emergency situations and permissive access control in emergency situations.

3) Security vs. communication range: Not long ago, IMDs did not support any means of wireless communication with external devices. Back then, IMDs were isolated and it was impossible for adversaries to conduct wireless attacks. However, this also prevented doctors from being able to collect telemetry data stored in the IMD or reprogram it once implanted. When the wireless communication was first introduced, the communication range between the device programmer and the IMD was only a few centimetres. This required the device programmer to be in close proximity to the patient's IMD for the entire duration of the communication, but it also limited the range of possible attacks. Since a few years, the device programmer and the IMD can communicate over a long-range channel. Even if this provides more flexibility to doctors and patients, extending the communication range also significantly increases the attack surface for IMDs.

4) Security vs. energy cost: While strong security mechanisms should be implemented in IMDs, they should also be lightweight to affect as little as possible the IMD's battery lifetime. The problem is that the level of protection provided by the security mechanisms is typically proportional to the energy required to execute them. For example, public-key cryptography could be suitable for IMDs because it offers strong protection and facilitates key management. However, even the most lightweight public-key primitive, elliptic curve Cryptography (ECC), is much more expensive than symmetric-key cryptography, specially in terms of communication cost. This does not mean that public-key cryptography cannot be used, but rather that its use should be limited.

It is important to note that the computation cost is typically negligible compared to the communication cost [92, 123]. Thus, one needs to primarily reduce the communication cost in order to decrease the total extra energy cost due to the use of cryptography. This means that, although a 128-bit Message Authentication Code (MAC) tag would be ideal for security reasons, this could be too costly for IMDs. Instead, a 64-bit tag could be used that does not have a significant impact on the IMD's battery at the cost of offering slightly less security.

All the aspects discussed above should be considered when designing countermeasures to preclude wireless attacks, while only some of them are relevant in the design of countermeasures that thwart analogue attacks.

4.2 Countermeasures for Wireless Attacks

We summarise the most prominent countermeasures for creating a secure wireless data exchange between a device programmer and an IMD. The main research challenge is how to perform key establishment. These solutions can be divided into five categories: (i) using an auxiliary or Out-Of-Band (OOB) channel, (ii) using pre-installed keys, (iii) using distance bounding protocols, (iv) using external devices and (v) using anomaly detection techniques.

Once a key is established between the device programmer and the IMD, it can be used to encrypt and authenticate the data transmitted over the wireless communication channel. However, even when wireless transmissions are encrypted, there are still some privacy risks remaining. For example, context information of the communication (e.g. the number of exchanged messages) could reveal sensitive information about patients. Countermeasures to defend against traffic analysis attacks are out of the scope of this thesis.

4.2.1 Auxiliary or Out-of-Band Channels

Various types of auxiliary or OOB channels have been proposed that allow to bootstrap a key establishment protocol between a device programmer and an IMD. We distinguish between the following three types of OOB channels: (i) using tattoos and bracelets, (ii) using audio or vibrational signals and (iii) using patient's physiological signals. Distance bounding protocols can also be considered an OOB channel but they will be discussed later in this section.

Tattoos and bracelets

One possible solution for establishing a key between a device programmer and an IMD – both in normal and emergency situations – would be to print a password in a bracelet worn by the patient or in his skin. However, bracelets can be lost, stolen or damaged and can implicitly reveal the patient's condition, which could lead to discrimination. Similarly, tattoos may become unreadable after an accident and could be refused by patients because of cultural, social or personal reasons, as shown by Denning et al. [55].

Audio and vibrational signals

Halperin et al. proposed a zero-power key distribution technique through which an IMD can generate a session key and transport it to a device programmer over an audio channel. Rather than using the IMD itself for generating and transporting the key, they proposed to use a passive RFID device, also known as Wireless Identification and Sensing Platform (WISP), which can be attached

to or built into an IMD. As the authentication process takes place between the device programmer and the WISP, this solution prevents DoS attacks as those proposed by Hei et al. [73]. The authors stated that the audio signals can only be received and demodulated by a device programmer in close proximity to the patient's IMD. However, Halevi et al. demonstrated that this solution is vulnerable to *acoustic eavesdropping attacks* where adversaries can recover the key by eavesdropping the acoustic emanations from several meters away [68].

Saxena et al. presented a pairing protocol by which a transmiter device can send a short PIN to a receiver device over a vibrational channel [119]. Their solution is simple and elegant but it would require to add a vibration motor or an accelerometer in the IMD. Furthermore, as vibration always results in unintentional acoustic emanations whose waveform is similar to the one of the vibration signal, this solution is also vulnerable to the *acoustic eavesdropping attacks* proposed by Halevi et al. [68].

Kim et al. [80] and Abhishek Anand et al. [38] introduced two similar vibration-based pairing techniques to defeat acoustic eavesdropping attacks. In both solutions, the transmitter device simultaneously sends a vibration signal and an audio signal. The former is used to transport the key, whereas the latter is generated using white noise and is used to hide the acoustic emanations from adversaries. Both solutions were shown to be effective at concealing the acoustic emanations to adversaries who eavesdrop the audio channel from far away. However, Abhishek Anand et al. showed that collocated adversaries who reside close to the transmitter device, can compromise the security of these solutions using standard signal processing and noise filtering techniques [39]. This is because white-noise-based audio signals cannot conceal the low-frequency (between 50-250 Hz) audio components, exposing the key to adversaries in close proximity to the source of the sound. Abhishek Anand et al. also proposed a novel technique for creating a masking signal that effectively masks both the low- and high-frequency audio components.

Collocated adversaries could be a real threat for IMDs since they could launch acoustic eavesdropping attacks through malicious wearable devices that are placed close to the implantation site. It is unclear if IMDs could produce the required low-frequency masking sounds and if sophisticated filtering techniques could be applied to separate the masking signal from the unintentional acoustic emanations produced by the vibration motor. As future work, it would also be interesting to investigate whether the vibration channel can be eavesdropped by collocated adversaries to recover the key.

Recently, Trippel et al. showed that the integrity of the readings taken by audio and vibration sensors can be compromised by injecting malicious analog accoustic signals [129]. As a result, both audio- and vibration-based key

transport techniques could be vulnerable to attacks where adversaries modify the key to a chosen value. This attack forces the IMD to establish the key with the adversary's device.

Physiological signals

Poon et al. were the first to propose the use of physiological signals extracted from the patient's body for establishing a cryptographic key between two devices [104]. In contrast to biometrics, which are person-specific and to some extent time-invariant, physiological signals are random signals that vary not only over time but also among individuals. Examples of physiological signals include the ElectroCardioGram (ECG), InterPulse Interval (IPI), PhotoPlethysmoGram (PPG), blood glucose, blood pressure, temperature, hemoglobin and blood flow [135]. However, only physiological signals that satisfy certain properties can be used to establish a key between a device programmer and an IMD. Below, we give an overview of all the properties that physiological signals should meet.

- **Readiness**: The process of acquiring the physiological signal should be as fast as possible.
- **Exclusivity**: Only the device programmer and the IMD should be able to accurately measure the physiological signal.
- **Availability**: The physiological signal should depend neither on the patient nor on his health condition.
- **Entropy**: The physiological signal should provide high entropy so that it is possible to generate random keys in short periods of time. More specifically, the physiological signal should offer high inter-subject variability and high intra-subject variability.
- **Precision**: The differences between the physiological signal measured by the device programmer and IMD should be as small as possible to maximise the Effective Key Length (EKL) and reduce the acquisition time.

In recent years, a significant number of papers have proposed to use the patient's IPI for generating random cryptographic keys. This is because the Least Significant Bits (LFBs) of the IPI seem to exhibit a high degree of entropy [114, 121]. Recently, Ortiz-Martin et al. conducted a large scale study to determine the suitability of the IPI as a source of entropy [101]. Their results pointed out that the patient's IPI may not provide sufficient randomness under some circumstances. Furthermore, there are several articles that have shown that it is possible to gather accurate information about the patient's IPI remotely [24,

47, 103, 122]. This renders all solutions that rely on the secrecy of the patient's IPI insecure[3].

A common approach for the device programmer and the IMD to agree on a key is that both devices take a measurement of the chosen physiological signal independently and synchronously. These measurements are typically not equal but at best only rather similar due to the noise. Furthermore, when using these measurements as a randomness source, the resulting key bits do not necessarily need to be uniformly distributed. The problem of how to generate cryptographic keys from noisy measurements has been widely studied in the literature (e.g. Dodis et al. [57] and Linnartz et al. [87])

Prior work in the domain of medical device security has proposed to use physiological signals in combination not only with fuzzy cryptographic primitives, as suggested by Dodis et al., but also with cryptographic commitment schemes. For a detailed overview of countermeasures proposed in the literature based on the use of the patient's physiological signal, we refer the reader to the fifth paper shown in Part II of this thesis.

Our contributions: (1) In the fifth paper of this thesis, we performed a critical evaluation of countermeasures by which an IMD can establish a key with a device programmer through the use of a patient's physiological signal. After a meticulous analysis of the existing countermeasures, we demonstrated that many of them rely on assumptions which do not hold in practical scenarios. In addition, we found serious security weaknesses in two pairing protocols proposed in the literature, namely the H2H [114] and the Biosec [49] protocols. The H2H protocol is shown to be vulnerable to a reflection and a Man-in-The-Middle (MiTM) attack. In the former, adversaries can successfully authenticate to the IMD simply by replaying the messages sent by the IMD, whereas in the latter the goal of the adversary is to trick the device programmer into believing that it is communicating with a legitimate IMD while it actually does it with the adversary. In the Biosec protocol, the authors did not account for the loss of entropy in the key due to the use of an Error Correction Code (ECC). Finally, we concluded our work by providing a set of recommendations on how to securely use physiological signals in cryptographic protocols.

(2) In [93], we explored the option of using a signal extracted from the patient's brain called Local Field Potential (LFP) as a source of randomness for generating cryptographic keys in neurostimulators. The choice of LFP is based on the fact that current neurostimulators can measure this signal and the hypothesis that

[3]The solution we introduced in [90] was proposed before knowing that the IPI does not provide enough entropy and that it can be measured remotely. Therefore, our solution can still be used if the IPI would be replaced by any physiological signal that satisfies the requirements previously mentioned.

it is impossible for adversaries to gather information about this signal remotely. Our preliminary results indicate that the bits extracted from the LFP signal are uniformly distributed. In addition to the proposal of a low-cost source of randomness, we introduced a novel technique for transporting the key from the neurostimulator to the device programmer. Our technique relies on applying an electrical signal with the key bits embedded in it to the neurostimulator's case. As both the neurostimulator's case and the patient's skin are conductive, any device programmer that can touch the patient's skin for several seconds, can measure this signal to recover the key. We validated our solution with some practical experiments.

4.2.2 Pre-installed Keys

Symmetric-key cryptography

Intuitively, a device-specific symmetric key could be pre-installed in each IMD during manufacturing and stored in a protected server. This would be a simple yet effective solution for non-emergency situations. In emergency situations, though, medical staff would need to contact the patient's doctor or the IMD's manufacturer on the fly for requesting the IMD's key. To overcome this limitation, Halperin et al. proposed to use a key diversification protocol where the IMD's key is generated through the IMD's serial number (SN) and a master key [70]. By storing the master key in all device programmers, medical staff could interrogate the IMD in situ to obtain its SN and generate the IMD's key. Nevertheless, this approach would bring substantial security risks since adversaries can easily obtain a device programmer either by stealing it from a hospital or by purchasing it in an auction site such as eBay. One possible way to slightly reduce the impact of compromising the master key would be to use several master keys instead of a single one. Another possibility would be to use a Physical Unclonable Function (PUF) or to store the master key in an encrypted form in tamper-resistant hardware. While this is a difficult and costly task, specially if tamper-resistant hardware is used, adversaries could attempt to recover the master key through a combination of side-channel and physical attacks. The incentives for adversaries to recover the master key are huge since compromising the key would put the security of millions of IMDs at risk.

Rather than storing the master key in the device programmer, one could opt for storing the master key in the cloud. As device programmers have an Internet connection, they could connect and authenticate to the cloud in order to derive the IMD's key. Once the communication with the IMD is finished, device programmers could delete the IMD's key from their memory not to leave

any trace to adversaries. The main limitation of this solution is that device programmers need to be able to operate at all times, including during Internet and cloud provider outages. Furthermore, this approach would still require to store a long-term key in all device programmers to authenticate to the cloud.

Public-key cryptography

Another possible solution would be to pre-install a public-private key pair in each IMD during manufacturing, and then use a secure key transport or key agreement protocol [36]. This would allow the device programmer and the IMD to establish a symmetric session key without having the key management issues previously mentioned. One could even argue that it is not necessary for IMDs to have a public-private key pair if the solution is solely based on public-key cryptography. IMDs only need to have the public keys of device programmers that may communicate with them, so that they can verify the authenticity of the messages sent by the device programmers. The main limitation of these approaches is that they would require to have a robust worldwide Public Key Infrastructure (PKI) that maintains an up-to-date list of trustworthy device programmers. In addition, IMDs should have a mechanism for updating and revoking keys such that device programmers can no longer communicate with them if they are compromised. Unfortunately, such a robust worldwide infrastructure is difficult to set up and maintain, and IMDs do not have an Internet connection to periodically get a Certificate Revocation List (CRL) or sufficient memory to store all the necessary certificates. Rather than using a PKI to authenticate public keys, one could use an authentication tree. The advantage of this approach is that it requires each device to store only its public key and the associated hash values from its public key to the root. However, every time a new public key is included in the tree, the root of the tree needs to be recomputed and new data needs to be stored in all devices.

Our contribution: In [91], we presented a novel key agreement protocol that relies on a master key stored in the device programmer, similarly to the solution proposed by Halperin et al. [70]. Our solution differs from theirs in that we propose a semi-offline (instead of offline) key agreement protocol where the master key is updated periodically. This means that adversaries who compromise the master key can only derive valid IMD keys for a limited period of time. We faced two important research challenges: (i) *how can device programmers and IMDs know when they need to use a new master key?* and (ii) *how can IMDs recompute their keys when a new master key is used?* For the former challenge, we leverage the fact that both device programmers and IMDs have a precise internal clock which is synchronised at every communication session. The latter challenge can be addressed by using bilinear pairings on the IMD. We argue that, even though this operation is expensive, it needs to happen only once in a while (e.g. every few months).

4.2.3 Distance Bounding Protocols

Distance bounding protocols allow to establish an upper-bound on the physical distance between two parties which are often denoted as verifier and prover. For this purpose, they rely on cryptography and physics. All distance bounding protocols proposed so far include at least two phases: (i) *a setup phase* and (ii) *a rapid-bit exchange phase*. In the former phase, both parties typically agree or commit to some parameters, whereas in the latter phase the verifier sends a series of single-bit challenges to which the prover replies with single-bit responses. To estimate the distance to the prover, the verifier measures the Round-Trip Time (RTT) between sending its challenge and receiving the response. In addition to the setup and rapid-bit exchange phases, some distance bounding protocols also include a *verification phase*.

The security of distance bounding protocols is based on the fact that it is impossible to transmit radio frequency (RF) signals faster than the speed of light. Additionally, the time required for the prover to process the challenge and compute the response should be negligible compared to the propagation time. If the verifier overestimates the prover's processing time (i.e. the prover can process the challenge faster), the protocol might become insecure since the prover could easily extend the distance from which it can successfully execute the protocol. However, if the verifier underestimates the prover's processing time, the prover may not be able to successfully execute the distance bounding protocol even when it is nearby.

Rasmussen et al. presented a proximity-based access control scheme suitable for pacemakers [109]. By using the Diffie-Hellman key agreement protocol in combination with distance bounding, a pacemaker can verify the distance and establish a session key with any device programmer that is in its close proximity. Instead of using an RF distance bounding, they proposed an ultrasonic distance bounding where the challenge and the response are transmitted over an RF channel and an audio channel, respectively. Two regions were defined with different security levels depending on the operation carried out by the device programmer. Critical operations, such as modifying the pacemaker settings, can only be conducted if the device programmer is at most 3 cm away. This distance condition is relaxed when performing non-critical operations, enabling the device programmer to communicate with the pacemaker as long as it is less than 10 m away.

The security of the solution of [109] relies on the fact that adversaries cannot transmit data on the audio channel using a signal that propagates faster than the speed of sound. The problem is that all ultrasonic distance bounding protocols are vulnerable to wormhole attacks where a MiTM adversary uses both a proxy-

prover and a proxy-verifier to convert the audio signal to a RF signal (and vice versa), as shown in Figure. 4.1 [120]. This allows to accelerate the transmission time on the relay channel, enabling adversaries to extend the distance from which they can successfully authenticate by several orders of magnitude. It is unclear, though, if this attack poses a real threat for IMDs. This attack could only be conducted by a malicious patient with an IMD while being in the doctor's office for a medical check up. He then forwards the messages it receives from the legitimate device programmer to an ally who is close to the victim. Next, the ally impersonates the legitimate device programmer by forwarding the messages it received.

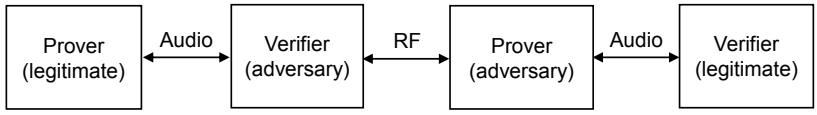

Figure 4.1: Wormhole attack.

The only way to protect against wormhole attacks is to use an RF distance bounding protocol. Nevertheless, designing a secure and practical RF distance bounding protocol that is suitable for IMDs is a very challenging task due to two main reasons.

Firstly, the device programmer would need to compute the response using an extremely fast function. Prior work has proposed different types of functions which can be divided into two main categories: (i) those executed in the digital domain (e.g. [46,71]) and (ii) those conducted in the analogue domain (e.g. [110]). Recently, a hybrid approach has also been proposed that leverages the benefits of the previous two approaches [108]. However, all of these solutions suffer from important limitations and downsides. When using a function executed in the digital domain, the processing time delay at the device programmer would be in the order of a few hundred nanoseconds. This delay could allow adversaries with dedicated hardware equipment to execute the protocol with the verifier from several meters away. On the contrary, when using a function executed in the analogue domain or an hybrid approach, complex hardware and storage would be needed at the device programmer. We argue that the latter would not be a problem since existing device programmers are powerful devices. Adding complex hardware and storage in device programmers would only slightly increase their cost.

Secondly, in this application the pacemaker takes the role of a verifier and the device programmer acts as a prover. Several articles have proposed distance

bounding protocols that are optimised for the prover. However, in all existing distance bounding protocols, the verifier is assumed to be a powerful device. A possible line of research for future work would be to design a distance bounding protocol that is optimised for the verifier and still protects against the classical distance bounding attacks.

4.2.4 External Devices

Multiple countermeasures have been introduced based on the use of an external device to mediate between the device programmer and the IMD. Table 4.1 shows a comparison between the existing solutions.

Denning et al. were the first to propose the use of an external device – which they called "cloaker" – to act as a bridge between the device programmer and the IMD [54]. This system is designed in such a way that it offers security in normal conditions and at the same time it allows doctors to access the patient's IMD in emergencies. If the cloaker is present, a secure wireless communication can be established between the cloaker and the IMD. When the cloaker is removed, the IMD can receive and respond to all messages. This leads to an important question: *how can IMDs know if the cloaker is present?* There are two possible ways to solve this challenge. In the first solution, the IMD remains in "sleep mode" until the cloaker starts the communication. The communication is initiated whenever the cloaker: (i) sends a "hello" message to announce its presence to the IMD or (ii) sends a message to inform the IMD that there is a legitimate device programmer which wants to communicate with it. The main limitation of this approach is that it is rather energy demanding for IMDs. This is because it would require IMDs to regularly wake up from "sleep mode" to check if there are incoming messages from the cloaker. In the second solution, the IMD queries the cloaker every time it receives a new message from a device programmer. This approach would be more energy efficient but at the same time it would expose IMDs to DoS attacks whose goal is to drain the IMD's battery. In both solutions, the cloaker authenticates the device programmer before the latter starts to communicate with the IMD.

There are several ways to handle the communication between the device programmer and the IMD when the cloaker is present. Similarly as before, each of these alternatives comes with its own advantages and disadvantages. The cloaker can either proxy the communication between the devices or provide the device programmer with a valid credential so that it can interact with the IMD directly. The former would allow the cloaker to keep track of all messages exchanged between the devices. However, this approach is rather demanding for the cloaker in terms of communications. On the contrary, the latter is less

demanding for the cloaker but it does not offer the possibility for the cloaker to track the messages exchanged between the devices.

In contrast to IMDs, the cloaker is an external device powered with a replaceable battery allowing it to perform complex operations. As a result, public-key cryptography can be used to secure the communication between the cloaker and the device programmer. A list of public keys of authorised device programmers can be pre-stored in the cloaker before it is given to the patient. Additionally, the cloaker could have an Internet connection so that it periodically downloads an up-to-date list of valid device programmer's public keys. However, having a robust worldwide PKI is costly and difficult.

For the cloaker and the IMD to establish a secure wireless channel, the authors suggested to use symmetric key cryptography. Nevertheless, this has important implications in terms of key management. As there is no mechanism to revoke or update keys in IMDs, if the cloaker is ever lost, damaged or stolen, the security of the patient's IMD could be at risk. It is also unclear whether storing keys in the cloaker provides a higher level of security than storing them in device programmers. Device programmers are typically used in controlled isolated locations, such as the doctor's office, whereas cloakers are worn by patients, making them more vulnerable to theft.

Table 4.1: Comparison between the existing countermeasures based on the use of external devices.

	Cloaker	IMDGuard	Shield
Requires to modify the IMD	Yes	Yes	No
Relies on pre-shared secrets with the IMD	Yes	No	No
Uses friendly jamming	Yes	No	Yes
Supports emergency mode	Yes	Yes	Yes
Known attacks	No	[113]	[128]

Xu et al. devised a security scheme called "IMDGuard" that is based on a wearable external device, also known as "Guardian", which acts as a bridge between the device programmer and the IMD. The Guardian performs an authentication process with the device programmer on the ICD's behalf [134]. In contrast to the cloaker, the Guardian does not rely on pre-shared secrets between the IMD and itself. Instead, the devices use the patient's ECG to establish a symmetric session key that is known only to them. This protects against adversaries who have permanent or temporal physical access to the Guardian. Nevertheless, similarly to the cloaker, the IMDGuard requires to modify the IMD and relies on public key cryptography for authenticating the device programmer. Once the device programmer is successfully authenticated,

the Guardian generates and provides a session key to the device programmer and the IMD. This allows these devices to communicate with each other directly (instead of exchanging messages through the Guardian). Following a similar approach as the one proposed by Denning et al. [54], the IMD switches to emergency mode if it does not detect the Guardian. This could be exploited by adversaries to conduct attacks that trick the IMD into believing that the Guardian is not present.

The IMDGuard offers security under the assumption that adversaries cannot make physical contact with the patient to measure his ECG. However, there are several papers that show that it is possible to gather information about the patient's ECG remotely without needing to physically touch the patient [24, 47, 103, 122]. Furthermore, Rostami et al. found that the IMDGuard is vulnerable to a MiTM attack which reduces its effective key length from 129 bits to 86 bits [113]. Although using cryptography with an 86-bit key would significantly improve the current situation, this research result shows that designing protocols for IMDs is a difficult task.

Gollakota et al. presented a small wearable external device, also known as "shield", that implements a full-duplex radio design that allows to simultaneously transmit and receive signals [67]. The shield acts as a proxy between the device programmer and the IMD, and ensures that no device programmer can communicate directly with the IMD. For this purpose, the shield uses friendly jamming to jam the messages to/from the IMD in order to prevent adversaries from decoding the messages, while still being able to decode them itself. The shield is proven to be very effective at protecting against adversaries who use the same power as the shield. For example, the shield can successfully jam all messages transmitted by this type of adversaries as long as they are more than 20 cm away from the IMD.

Similarly to the previous solutions, the shield uses standard cryptographic mechanisms to establish a secure channel with the device programmer, and can be removed or turned off in emergencies. However, unlike previous solutions, it does not require to modify the IMD. Its main limitation is that it cannot protect against adversaries who transmit signals with more power than the jamming signals transmitted by the shield. For example, adversaries with 100 times the shield's power can successfully send messages to the IMD from distances up to 5 meters. Furthermore, Tippenhauer et al. showed that it is possible to perform attacks that compromise the message confidentiality [128]. By placing several antennas in a specific set of locations, adversaries could suppress the jamming signal and recover the data sent by the devices. However, this attack does not break the authentication requirement, which is much more important than the confidentiality requirement.

From a research point of view, the shield offers an interesting and effective way of mitigating the existing security issues, and is specially suitable for IMDs that will remain within the patient's body for several more years. Nevertheless, an important drawback of the shield is that jamming could disrupt ongoing communications between other legitimate devices. In some countries, jammers are illegal and their use can result in large fines. We believe it is very unlikely that this type of solutions will be accepted by the FDA and hence adopted by IMD manufacturers.

4.2.5 Anomaly Detection

Existing solutions based on anomaly detection leverage the fact that IMDs typically communicate to a small number of device programmers at specific locations and pre-determined times.

Hei et al. introduced a lightweight security scheme based on the use of access patterns to the patient's IMD and on machine learning [73]. Their solution is intended (i) to defend only against *resource depletion attacks* and (ii) to be used in combination with other security mechanisms to restrict access to the IMD only to legitimate device programmers. For each individual IMD, detailed information about the patient/doctor's access pattern is first collected. This data is then used as a training data in a Support Vector Machine (SVM) classifier to create a model. The authors suggested that the patient's cell phone could store this data and run the classification algorithm.

Every time the IMD receives a new message from a device programmer, the IMD notifies the patient's cell phone, which in turn runs the classification algorithm. If the patient's cell phone decides that the device programmer is not legitimate, it sends a message to which the IMD immediately switches to sleep mode for a specific amount of time. This makes it more difficult for adversaries to launch resource depletion attacks. If the output of the classifier is uncertain, the decision of accepting the device programmer as legitimate is left to the patient.

This solution presents several limitations. Firstly, it provides only very weak security protection. While strict policies can increase the detection rate of attacks, they could also result in an increased number of false alarms. Secondly, the authors did not describe how the patient's cell phone and the IMD can securely establish and manage cryptographic keys. This task becomes even more difficult due to the lack of mechanisms to update and revoke keys in IMDs and the possibility that the patient's cell phone gets stolen or lost. Thirdly, this solution is neither lightweight nor flexible. Finally, adversaries can leverage that the device programmer can authenticate to the IMD only after it has been

verified by the solution proposed by Hei et al., to perform DoS attacks. These attacks can be launched to prevent access to the patient's IMD when needed. To achieve their goal, adversaries could jam the messages from the patient's cell phone to the IMD.

Zhang et al. proposed "MedMon", a multi-layered anomaly detection monitor that can be implemented in a dedicated external device or in the patient's phone [136]. To determine whether transmissions are legitimate, MedMon relies on finding anomalies in the signals sent by device programmers (i.e. physical anomalies) as well as in the patient's behaviour (i.e. behavioural anomalies). To this end, MedMon first measures several characteristics of the physical signal, such as the Received Signal Strength Indicator (RSSI) or the Time Of Arrival (TOA), and compares them with the ones obtained during the training phase. If these values are within a certain threshold, MedMon proceeds to find behavioural anomalies. As the actions performed by patients typically follow a pattern – unless it is an emergency – MedMon also inspects the data contained in the messages and measures the frequency of certain operations. Similarly as before, it compares these values with the ones from the training phase and accepts the message as valid if they are similar. Otherwise, MedMon can trigger several actions depending on the predefined security policy. This includes passive actions, such as notifying the user by emitting a sound, or active actions, such as jamming the malicious messages before they reach the IMD.

The main advantage of MedMon is that it is suitable for legacy devices since it does not require to modify the IMD. Nevertheless, MedMon has several disadvantages. Firstly, MedMon does not provide strong security protection. Adversaries could bypass both the "physical" and "behavioural" security checks to modify the IMD's settings. In addition, MedMon opens the door to DoS attacks where adversaries could repeatedly send messages to the IMD to perform a certain action so that the IMD prevents legitimate device programmers from performing this action when necessary. Secondly, MedMon does not offer message confidentiality. Passive adversaries who eavesdrop the wireless channel can intercept the messages to/from the IMD in order to learn patient private information. Finally, MedMon needs to be trained for each specific patient and always needs to be worn in the same location (e.g. the patient's wrist). It is unclear what the effect is of slightly modifying the location of MedMon. The authors did not discuss if this could prevent legitimate device programmers from communicating with the IMD, or could allow malicious messages to reach the IMD.

4.3 Countermeasures for Analogue Attacks

There exist several countermeasures to detect or mitigate signal injection attacks on sensors based on intentional, low-power EMI signals [84]. These countermeasures aim at finding ways of (i) attenuating the injected signal on the sensor's analogue circuit, (ii) differentiating between the injected and the measured signal and (iii) removing the injected signal. The proposed countermeasures range from *hardware-based (or analogue) defences*, such as shielding, differential comparators and filters, to *software-based (or digital) defences* such as signal contamination, adaptive filtering, cardiac probe and reverting to a safe default. Most of the hardware-based defences have been already implemented in IMDs. However, they suffer from important limitations. For example, IMDs cannot use very selective filters since this could also attenuate the real measured signal. In addition, hardware-based defences may not be sufficient to protect against some types of attacks.

Several software-based defences were proposed to defeat against adversaries who are equipped with a strong emitter or conduct baseband attacks. For example, they proposed a countermeasure based on anomaly detection that measures the EMI level in the environment to determine if the sensor is under attack. Similarly, adaptive filtering allows to attenuate the EMI in order to increase the Signal-to-Interference (SI), whereas active probing leverages the direct connection between the IMD and the cardiac tissue to distinguish between the injected and the intended signal.

More research is needed to determine if all these countermeasures (i) can be implemented in IMDs in practice or (ii) can affect the real measured signal in some circumstances. All countermeasures should be designed considering the constraints of IMDs and tensions explained in Section 4.1.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

An important part of our work focused on analysing the security of proprietary protocols used by commercial IMDs to wirelessly communicate with device programmers. Our research has shown that IMD manufacturers rely on proprietary protocols with limited or no security protection as their only means to provide "security". There could be several reasons why manufacturers omitted security on IMDs. We believe that this can be due to the lack of awareness and expertise, the extra cost that this introduces or the difficulty to guarantee that security does not affect safety and availability. We have demonstrated that proprietary protocols can be reverse-engineered and consequently that security-through-obscurity is a dangerous approach that should never be used. The reverse engineering of these protocols led us to perform passive and active wireless attacks against several types of IMDs which could compromise the safety and privacy of patients. It is important that IMD manufacturers migrate from the current insecure proprietary closed solutions to open and thoroughly scrutinised security solutions.

Another important aspect of our work was the proposal of countermeasures for addressing the security issues caused by wireless attacks. For designing practical countermeasures, we had to consider the important constraints of IMDs as well as the inherent tensions between some of the IMDs' functional requirements and the desirable security goals. In this line of work, our first research question was: *is it possible to use cryptography in IMDs without significantly reducing their battery lives?* To answer this question, we evaluated the feasibility of

cryptography in a widely used insulin pump system. Our work demonstrated that data confidentiality can easily be achieved through an optimised message format, whereas data authentication is difficult to accomplish without negatively affecting the insulin pump battery. Our work pointed out that there is a need for future research on how to protect the message authentication without significantly increasing its length.

We have also investigated several ways how to establish and manage cryptographic keys between a device programmer and an IMD. More specifically, we tackled this problem in two different ways depending on whether the devices have pre-installed keys. For the former case, we proposed a novel semi-offline key agreement protocol that limits the time that a master key can be used to derive the IMD's key. For the latter case, we presented a novel security architecture that includes the secure generation and transportation of keys from the IMD to the device programmer.

There exist a significant number of countermeasures to mitigate or solve wireless attacks on IMDs. However, most of these solutions suffer from limitations and downsides, or have been broken by us and other researchers. We have performed a critical evaluation of the most prominent countermeasures that rely on patient's physiological signals for establishing a key between the device programmer and the IMD. This analysis resulted in the identification of serious protocol and implementation weaknesses in two protocols proposed in the literature. We also evaluated several assumptions on the adversaries and showed that some of them do not hold in practical scenarios. Finally, we provided a set of security design guidelines where we detailed the steps to be followed in order to use patient's physiological signals in cryptographic solutions.

Cryptography is needed not only to create a secure wireless channel between the device programmer and the IMD, but also to protect the communication links between other devices within the system. This also includes developing security mechanisms for concealing the context information of these communications, which is also known as metadata. This is of utmost importance since meta-data could reveal highly sensitive information about patients. In this line of work, we have presented a protocol through which an IMD can establish an end-to-end secure channel with a hospital for remote monitoring and reprogramming of the IMD while the patient is at home. An important feature of our protocol is that it has been designed to protect the patient's privacy. More specifically, our protocol prevents unauthorized entities and adversaries from discovering the patients' real ID, their location or being able to link their messages.

5.2 Future Work

The field of security and privacy of medical devices has received significant attention over the last few years. However, this field still has many open research problems. This section gives an overview of potential research directions for future work on the security and privacy of IMDs.

1) Unified framework for analysing the security of IMDs

- **Security analysis of new medical devices**. In the upcoming years, manufacturers will start to implement cryptographic solutions in the new generations of IMDs. A potential direction for future work could be to investigate the security of these cryptographic mechanisms. Adding security mechanisms into IMDs is challenging. This implies not only using well-scrutinised cryptographic primitives and protocols, but also finding ways to appropriately manage cryptographic keys and protect the devices against physical attacks. Therefore, a non-exhaustive list of aspects that could be explored include: (i) identifying design and/or implementation weaknesses in the cryptographic protocols, (ii) recovering cryptographic keys from device programmers or base stations by performing side-channel or physical attacks, or (iii) reverse engineering and analysing the security of ciphers in case they are proprietary.
- **Developing tools for automated black-box protocol reverse engineering**. Reverse engineering a proprietary protocol is a laborious manual process that is often specific to a particular type of device, model or manufacturer. One possible direction for future work could be to design a tool that automatically reverse engineers protocols given some messages exchanged between the device programmer and the IMD over the air. Recently, several solutions have been proposed to automate the process of reverse engineering protocols based on the use of machine learning [51]. However, it is unclear what the efficacy of these solutions is in determining the message format and protocol state machine for proprietary protocols used by IMDs.
- **Analogue attacks**. As IMDs become more computationally complex and interconnected, they will also be more exposed to other types of cyber attacks. Recently, researchers have shown that analogue attacks can be conducted against IMDs by compromising the integrity of sensors readings. However, the proposed attacks can only be conducted from a distance up to 2 cm from the IMD, which significantly limits their impact in practice. As a future work, researchers could attempt to overcome these limitations and investigate how to perform these attacks from further away or if these

low-power EMI could be produced by wearable devices that the patient could carry close to the implantation site.

- **Software attacks**. Sooner than later, IMDs will be connected to the Internet. This will allow doctors in the hospital to remotely (i) reprogram the patient's IMD while the patient is at home and (ii) update the IMD's software remotely. However, all these advantages come at the cost of significantly broadening the attack surface of IMDs. For example, this would make these devices more vulnerable to software vulnerabilities and malware. Studying whether it is possible to inject malicious software into IMDs could be an interesting avenue for future work.

2) Unified framework for evaluating countermeasures for IMDs

- **Analogue defences**. There is a need for designing simple yet effective countermeasures that help defending against attacks that aim at compromising the integrity of sensors' readings. To achieve this, a combination of defences – both in hardware and software – is required. In order to evaluate the suitability of the defences, it is important not only to investigate their complexity and cost but also to ensure that they do not affect the real measured signal. For example, redundancy could potentially help to detect attacks but this may not be effective due to the limited size of the IMD. However, it is still unclear which would be the most interesting research direction to tackle this problem.

- **Software defences**. Research is needed to ensure that IMDs do not execute malicious software. To this end, there is a need to develop mechanisms to detect malicious software and quarantine it before it is executed. However, this is a challenging task because these devices have limited memory storage and do not have an Internet connection that enables them to be aware of the most recent security threats. An interesting direction for future work would be to develop a lightweight runtime anomaly detection system that reveals suspicious behaviour of the IMD.

Part II

Publications

List of publications

International Journals

1. Eduard Marin, Enrique Argones Rúa, Dave Singelée, and Bart Preneel, "A critical evaluation of physiological-signal-based solutions for medical devices," (Currently under submission), 2018.

International Conferences/Workshops

1. Eduard Marin, Dave Singelée, Bohan Yang, Vladimir Volskiy, Guy Vandenbosch, Bart Nuttin, and Bart Preneel, "Securing wireless neurostimulators," In ACM Conference on Data and Application Security and Privacy (CODASPY 2018), 12 pages, 2018.
2. Tomer Ashur, Jeroen Delvaux, Sanghan Lee, Pieter Maene, Eduard Marin, Svetla Nikova, Oscar Reparaz, Vladimir Rozic, Dave Singelée, Bohan Yang, and Bart Preneel, "A Privacy-Preserving Device Tracking System Using a Low-Power Wide-Area Network (LPWAN)," In Cryptology and Network Security, International Conference (CANS 2017), 22 pages, 2017.
3. Pieter Robyns, Eduard Marin, Wim Lamotte, Peter Quax, Dave Singelée, and Bart Preneel, "Physical-Layer Fingerprinting of LoRa devices using Supervised and Zero-Shot Learning," In Proceedings of the 7th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec 2017), 6 pages, 2017.
4. Eduard Marin, Dave Singelée, Flavio D. Garcia, Tom Chothia, Rik Willems, and Bart Preneel, "On the (in)security of the Latest Generation Implantable Cardiac Defibrillators and How to Secure Them," In Annual Computer Security Applications Conference (ACSAC 2016), 10 pages, 2016.

5. Aysajan Abidin, Eduard Marin, Dave Singelée, and Bart Preneel, "Towards quantum distance bounding protocols," In Workshop on RFID Security (RFIDsec 2016), Lecture Notes in Computer Science, Springer-Verlag, 14 pages, 2016.

6. Eduard Marin, Mustafa. A. Mustafa, Dave Singelée, and Bart Preneel, "A Privacy-preserving Remote Healthcare System Offering End-to-End Security," In Proceedings of the 15th International Conference in Ad-hoc, Mobile and Wireless Networks (ADHOC-NOW 2016), 14 pages, 2016.

7. Eduard Marin, Dave Singelée, Bohan Yang, Ingrid Verbauwhede, and Bart Preneel, "On the Feasibility of Cryptography for a Wireless Insulin Pump System," In ACM Conference on Data and Application Security and Privacy (CODASPY 2016), 8 pages, 2016.

National Conferences

1. Karel Domin, Eduard Marin, and Iraklis Symeonidis, "Security Analysis of the Drone Communication Protocol: Fuzzing the MAVLink protocol," In Proceedings of the 37th Symposium on Information Theory in the Benelux, 7 pages, 2016.

2. Eduard Marin, Sophie Pollin, and Dave Singelée, "Security Analysis Of An Implantable Cardioverter Defibrillator," In Proceedings of the 34th Symposium on Information Theory in the Benelux, 7 pages, 2013.

Internal Reports

1. Eduard Marin, Dave Singelée, and Bart Preneel, "Attacks in H2H protocol," COSIC internal report, 3 pages, 2016.

2. Eduard Marin, Dave Singelée, and Bart Preneel, "Security evaluation of Biosec," COSIC internal report, 6 pages, 2016.

3. Eduard Marin, Dave Singelée, and Bart Preneel, "Secure remote reprogramming of implantable medical devices," COSIC internal report, 7 pages, 2014.

Publication

A Privacy-preserving Remote Healthcare System Offering End-to-End Security

Publication Data

Eduard Marin, Mustafa. A. Mustafa, Dave Singelée, and Bart Preneel, "A Privacy-preserving Remote Healthcare System Offering End-to-End Security," In Proceedings of the 15th International Conference in Ad-hoc, Mobile and Wireless Networks (ADHOC-NOW 2016), 14 pages, 2016.

Contributions

- Principal author. The protocol design as well as the security and privacy analysis is the result of joint work with co-authors.

A Privacy-preserving Remote Healthcare System Offering End-to-End Security

Eduard Marin, Mustafa A. Mustafa, Dave Singelée, and Bart Preneel

ESAT-COSIC and iMinds, KU Leuven, Belgium

Abstract. Remote healthcare systems help doctors diagnose, monitor and treat chronic diseases by collecting data from Implantable Medical Devices (IMDs) through base stations that are often located in the patients' house. In the future, these systems may also support bidirectional communication, allowing remote reprogramming of IMDs. As sensitive medical data and commands to modify the IMD's settings will be sent wirelessly, strong security and privacy mechanisms must be deployed.

In this paper, we propose a user-friendly protocol that is used to establish a secure end-to-end channel between the IMD and the hospital while preserving the patient's privacy. The protocol can be used by patients (at home) to send medical data to the hospital or by doctors to remotely reprogram their patients' IMD. We also propose a key establishment protocol between the IMD and the base station based on a patient's physiological signal in combination with fuzzy extractors. Through security analysis, we show that our protocol resists various attacks and protects patients' privacy.

1 Introduction

Remote healthcare systems usually collect data from medical devices implanted within the patient's body, also known as Implantable Medical Devices (IMDs), several times per day through a base station that is often installed in the patient's house. In the near future, these systems may support bidirectional communication to enable remote reprogramming of the IMD by a doctor in a hospital. While these remote healthcare systems can improve the patients' quality of life and extend the time they can live independently at home, they pose important security and privacy risks. Currently, proprietary protocols are being deployed with limited security measures [70, 92]. These insecure wireless

protocols may lead to several attacks that can result in fatal consequences for patients.

Remote healthcare systems are typically built such that there is a central node (denoted as data concentrator in the rest of this paper) that collects and redirects all (encrypted) data sent between base stations and hospitals. Context information about these communications, i.e. metadata, can be valuable for insurance companies or government agencies to monitor public health or compile statistics. However, even if cryptography is used, metadata can leak patients' sensitive information to the data concentrator, which may lead to abuse of data, e.g. denying individuals an insurance contract. To mitigate some of these issues, a trivial solution would be to have several data concentrators (instead of only one) that do not cooperate with each other. This solution would be expensive to deploy and difficult to maintain. In addition, there would not be any guarantee that the data concentrators do not share data with each other. Another solution would be to use a fresh pseudonym in every message. Although this approach would make it more difficult for adversaries to compromise the patient's privacy, this is not sufficient; if the data concentrator knows where the base station is located, then unique patient identifiers may be revealed.

Our first contribution is a user-friendly protocol that allows an IMD to establish an end-to-end secure channel with a hospital while preserving the patient's privacy. The protocol can be used by patients (at home) to send medical data to the hospital or by doctors to remotely reprogram the IMD of their patients. Our protocol makes use of cryptography and an anonymous communication channel to prevent the data concentrator from learning patients' sensitive information. Our second contribution is a key establishment protocol that allows an IMD and a base station to agree on a symmetric session key without using public-key cryptography or any pre-shared secrets between devices. Instead, our protocol uses a patient's physiological signal in combination with fuzzy extractors.

2 Related Work

2.1 Security and Privacy in Remote Monitoring Systems

While much research has focused on making remote monitoring systems more reliable, unobtrusive, energy efficient and scalable, security and privacy have received less attention [88, 98]. Ko et al. acknowledged the importance of security and privacy, but they did not provide details about cryptographic mechanisms [81]. Ortiz et al. proposed a protocol to secure the wireless communication between devices in a medical system [100]. However, the

protocol does not provide message integrity, and the receiver keeps a list of keys for each transmitter. Transmitters send messages along with their unique device identity (ID) unencrypted to indicate to the receiver which of the keys is used to decrypt the message. This may result in a privacy breach since adversaries can use the unique transmitter ID to track, identify or locate individuals. Perrig et al. presented SNEP, a protocol that provides data confidentiality, integrity, mutual authentication and message freshness [102]. Pre-installed symmetric keys, shared between each device and the base station, are used to derive a new session key every time a device starts communicating with the base station. In contrast to Perrig et al., we also consider privacy. More specifically, our protocol aims to prevent unauthorized entities and adversaries from discovering the patients' real ID, their location or being able to link their messages. Furthermore, the devices are implanted within the patient's body and the base stations do not have any pre-shared keys with them, which makes key management more challenging.

2.2 Key Establishment Protocols

The unique characteristics of IMDs pose novel challenges in the design of key establishment protocols and key management solutions. IMDs are battery-powered and resource-constrained devices in terms of size, memory, processor and energy. The battery typically lasts 7 years. When the battery is drained, surgery is needed to replace the IMD. We note that a trade-off between security and open-access in emergencies is also required. Consider a cardiac patient who is travelling. Although it is clear that no one should be able to access his pacemaker while he is walking on the street, medical staff should have immediate access to his IMD in an emergency situation.

Intuitively, one possible way for the IMD and the base station to establish a key would be to use public-key cryptography. However, IMDs cannot use expensive cryptographic primitives in terms of computations and power consumption. Another possibility would be to pre-install a master (symmetric) key in all IMDs, but this may prevent a patient from receiving care in an emergency situation. A pairing protocol could also be used for establishing a symmetric session key [48, 66, 127]. Nevertheless, IMDs do not contain a screen, a keyboard or an accelerometer and it is not possible to physically access them once implanted; thus none of the existing solutions can be used. In this paper, we propose a key establishment protocol in which the IMD and the base station measure a patient's physiological signal independently and synchronously to agree on a symmetric key. The protocol uses the time between heart beats, also known as InterPulse Interval (IPI), as the source of randomness, similarly to the touch-to-access protocol proposed by Rostami et al. [114]. Unlike their work, our

protocol is more efficient as it only uses symmetric cryptography in combination with fuzzy extractors.

3 Design Preliminaries

3.1 System Model

Our remote healthcare system, similar to existing architectures, consists of the following entities (see Fig. 1). Without loss of generality, in the rest of this paper we will assume that the IMD is a pacemaker.

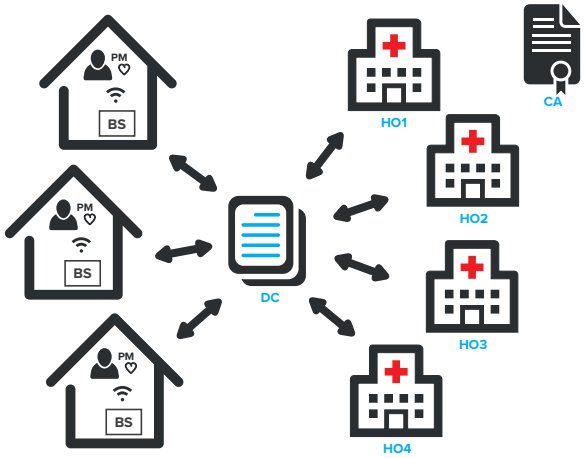

Figure 1: Our remote healthcare system comprises pacemakers (PMs), base stations (BSs), a data concentrator (DC), hospitals (HOs) and a certification authority (CA).

A *pacemaker* (PM) is a device implanted within a patient's body that is used to monitor and control his heart beat. A *base station* (BS) is an external device installed in a fixed location (e.g. home or a hotel) which collects and forwards medical data to a hospital, and sends commands to PMs as instructed by a doctor in the hospital. BSs have a programming head that incorporates a built-in sensor to read a patient's physiological signal (e.g. the IPI). A *data concentrator* (DC) acts as a bridge between BSs and hospitals, and is in practice often managed by the company that manufacturers the PMs and BSs. A *hospital* (HO) is a medical institution where doctors analyse the medical data and send commands to PMs, whereas a *certification authority* (CA) is a trusted entity that issues digital certificates to HOs and the DC.

A BS and a PM can communicate with each other wirelessly using the Medical Implant Communication Service (MICS) band [117], or any other low-energy wireless technology. The communication range between the BS and the PM depends on the wireless communication technology being used, e.g. from two to five meters when using the MICS band. The communication between the BS and the DC takes place over the Internet using a low-latency anonymous communication channel (e.g. a Mix network [118]), whereas the communication between the DC and a HO takes place over the Internet using a standard secure channel, e.g. TLS. To balance the load, multiple DCs are in place; however, for the sake of simplicity, we consider them as one in the rest of this paper.

We consider two possible scenarios depending on whether the doctor is on-line or off-line. For an on-line remote medical check, the patient first makes an appointment with the doctor (e.g. via telephone). At the time of the appointment, the patient sends medical data to the HO through the base station. The doctor then analyses the data and, if required, sends commands to the PM. If the doctor is off-line, the patient can still send medical data to the HO; however, these data will be processed by the doctor at a later stage. We note that in our system the doctor can only reprogram the patient's PM in the on-line scenario.

3.2 Threat Model and Assumptions

Threat Model: PMs and HOs are honest and trusted. PMs follow the protocol specifications as designed by their manufacturers and the U.S. Federal Communications Commission (FCC). PMs can only establish one communication session with a BS at a time, and do not initiate any communication without receiving a request from a legitimate BS [6]. Adversaries can eavesdrop, modify, inject and jam the messages exchanged between any of the entities. Adversaries can observe only a fraction of the Internet network traffic. This is a common assumption when using low-latency anonymous communication systems. In addition, adversaries might compromise any number of BSs, including the one being used by the patient. The DC is honest but curious; it follows the protocol specifications but it might attempt to discover information about patients by looking at meta-data.

Assumptions: We assume that adversaries cannot make physical contact with the patient without this being noticed by the patient. We assume that the communication between the BS and the DC takes place over the Internet using an anonymous communication channel. However, we do not specify which type of anonymous channel is used, since this is out of the scope of this paper. We assume that all entities (except PMs) have the CA's certificate pre-installed. We

assume that each HO has a server with a database that contains a list of their patients along with their corresponding PM IDs and cryptographic keys. The server is located in a secure room where it cannot be stolen or tampered with; only authorized medical staff has access to it through appropriate identification, authentication and authorization mechanisms.

3.3 Design Requirements

Our remote healthcare system should satisfy the following functional (F), security (S) and privacy (P) requirements.

(F1) User-friendly: Reporting medical data and reprogramming the PM should be easy and convenient for patients.

(F2) BS-independent: Patients should be able to use any legitimate BS, even the ones belonging to other patients.

(F3) Energy Cost: Computational cost at PMs should be as low as possible to reduce the energy consumption.

(S1) Mutual Entity Authentication: PMs and HOs should be assured of each other's identity when receiving messages.

(S2) Message Integrity: PMs and HOs should be assured that the received messages are fresh and have not been altered during transit.

(S3) Confidentiality of Medical Data and Commands: Only the authorized HO should be able to access patient's medical data and send commands.

(S4) Availability: Ensures that the system is accessible upon demand by authorised entities.

(P1) Patient Privacy (minimum data disclosure):

(P1.1) Patient Identity Privacy: Only HOs should know the identity of the patient who sends medical data.

(P1.2) Hospital Identity Privacy: No unauthorized entity should know to which hospital the patient sends medical data.

(P1.3) Location Privacy: No entity should infer the patient's location [1].

[1] In an emergency situation doctors have other means (e.g. necklace-based emergency systems) to know the patient's location.

(P1.4) Session Unlinkability: Only the HO where the patient is registered should be able to link messages sent from the same patient.

4 The Protocol

This section presents our protocol for medical data reporting and remote reprogramming of the patient's PM. It provides end-to-end security (i.e. data confidentiality, integrity, mutual authentication and message freshness) between the patient's PM and the hospital, and protects the patient's privacy. Our protocol is divided into two stages: the medical data reporting stage and the PM reprogramming stage. Prior to the detailed protocol description, we first outline the system initialization process. Table 1 shows the notation used in the paper.

4.1 System Initialisation

Each HO and the DC generate a public/private key pair, PK_{ho_i}/SK_{ho_i} and PK_{dc}/SK_{dc}, respectively. The public keys are signed by the CA. A list of valid HO certificates is stored in the DC, whereas each HO has a valid DC certificate. A group signature scheme is used by BSs to anonymously sign messages on behalf of the BSs group, so that the DC can still verify the authenticity of the message without knowing which specific BS signed the message, thus hiding the ID and location of the patient. All BSs use the same (group) public key, $PK_{bs_{group}}$, and have distinct private keys, SK_{bs_i}. Next, the DC signs the BSs group public key, generates a digital certificate that contains the ID and group's public key, and stores it. The certificate of the DC is pre-installed in all BSs.

During the PM manufacturing process, a symmetric key, $K_{ho\text{-}ps}$, is pre-installed in each PM. $K_{ho\text{-}ps}$ is shared between all PMs and the DC, and used for generating HO pseudonyms. Our protocol uses the same $K_{ho\text{-}ps}$ for all PMs, so that the DC cannot identify the PM that generated the HO pseudonym. In the PM setup phase, two independent symmetric keys, $K_{pm\text{-}ho}$ and $K_{pm\text{-}ps}$, are generated and installed in each PM. The procedure takes place in the HO before the PM is implanted to avoid the PM's manufacturer (often the DC) from learning these keys. $K_{pm\text{-}ho}$ and $K_{pm\text{-}ps}$ are shared between the PM and its corresponding HO. The former is used for encrypting the patient's medical data whereas the latter for generating PM pseudonyms.

Table 1: Notations.

Symbols	Meanings
d, cmd	medical data of a patient, command sent to the PM
ID_i, PS_i	unique identity of entity i, pseudonym of entity i
$\mathrm{K}_{\text{pm-ho}}$	key shared between PM and HO to encrypt/decrypt data/commands
$\mathrm{K}_{\text{pm-ps}}$	key shared between PM and HO to create PM pseudonyms (ps)
$\mathrm{K}_{\text{ho-ps}}$	key shared between all PMs and DC to generate HO's pseudonyms (ps)
K_{s}	session key established between PM and BS
N_i, TS_i	nonce generated by entity i, timestamp produced by entity i
$\mathrm{msg}_{i\text{-}j}$	message constructed by entity i and intended for entity j
$\mathrm{ct}_{i\text{-}j}$	counter used in messages between the entity i and the entity j
$\mathrm{C}_{i\text{-}j}$	ciphertext generated by entity i and intended for entity j
PK_i, SK_i	public and private key of entity i
$PRF_{\mathrm{K}}(\mathrm{M})$	pseudorandom function of message M with key K
$AE_{\mathrm{K}}(\mathrm{M})$	authenticated encryption of message M with key K
$E_{\mathrm{PK}_i}(\mathrm{M})$	asymmetric encryption of message M with public key of entity i
$\mathrm{Sig}_i(\mathrm{M})$	digital signature of entity i on message M

Various circumstances may cause the certificates to become invalid before their expiration date. If the private key of any of the entities is compromised, a new public/private key pair is generated (as explained above). The new public key is then signed by the CA and broadcasted to the network. All entities can then verify the message authenticity by using the CA's public key. From that point onwards, the old certificate is no longer valid. If the private key of any BS is compromised, the BS is sent to its manufacturer for being reconfigured and replaced. We note that group signature schemes typically allow revocation and addition of new members (i.e. BSs) into the group (for more details, see [44]).

4.2 Medical Data Reporting Stage

After the system initialisation phase, the PM can report medical data to the HO (see Fig. 1). Prior to each reporting stage, a new symmetric session key is established between the PM and the BS for securing the data exchanged over the air. We next describe the proposed key establishment protocol followed by the actual reporting stage more in detail.

IPI-based Key Establishment Protocol: Our protocol requires the IMD and the BS to independently measure the patient's IPI (i.e. time between heart beats) at the same time. These IPI readings, which can be measured

anywhere in the patient's body just by touching the patient's skin, are then used for establishing a symmetric key that is valid only for one session. Previous work has shown that the four least significant bits of IPIs are uncorrelated and independently identically distributed (i.i.d) [134]. The IPI cannot be read remotely (e.g. via a webcam), as shown by Rostami et al. [114]. Therefore, based on their results and our assumptions, a remote attacker cannot measure the IPI. Fig. 3 shows the IPI-based key establishment protocol.

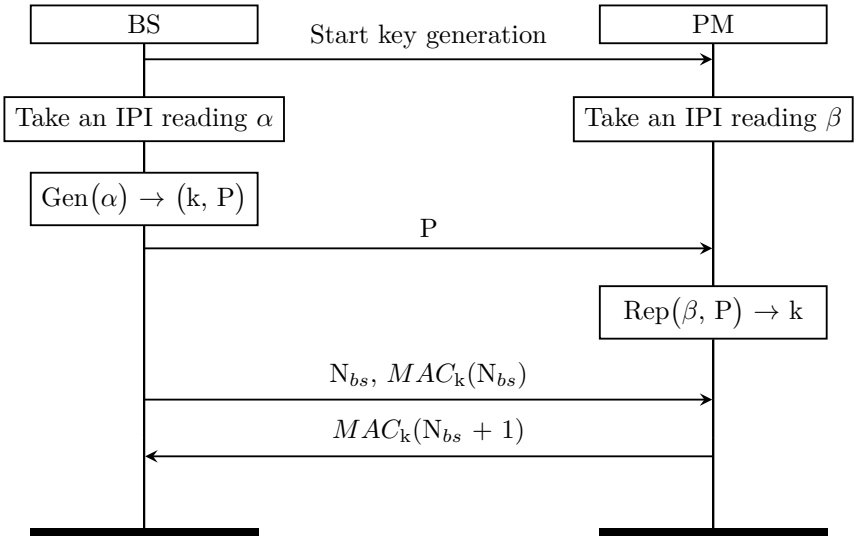

Figure 2: IPI-based key establishment protocol between the BS and the PM.

To trigger the key establishment procedure, the patient first presses a button on the BS. The PM and BS then take two readings of the patient's IPI at the same time. However, these readings are not equal (but rather similar) due to the noise. Let us denote the reading taken by the BS as α and the reading taken by the PM as β. Fuzzy extractors allow generating a cryptographic key k from α and then successfully reproduce k from β, iff α and β are almost equal [57]. Fuzzy extractors are composed by two functions: generate $\big(Gen\big)$ and reproduce $\big(Rep\big)$. *Gen* is executed by the BS with α as an input, and outputs a key k $\epsilon \left\{0,1\right\}^{l}$ and helper data P $\epsilon \left\{0,1\right\}^{*}$. The BS then sends P in order to help the PM to reproduce k. The PM executes *Rep* with β and P as inputs, and outputs a key *k'* (if β and α are similar, then k equals *k'*).

To achieve key confirmation, the BS generates a nonce, $\mathrm{N_{bs}}$, and sends it to the

PM along with a Message Authentication Code (MAC), $MAC_{\mathrm{k}}(\mathrm{N_{bs}})$. Upon receiving the message, the PM verifies $MAC_{\mathrm{k}}(\mathrm{N_{bs}})$ using *k'*. If the MAC is verified correctly, the PM is assured that the BS knows the key; however the BS does not have any assurance that the PM knows the key. For the PM to prove knowledge of the key, it computes $MAC_{\mathrm{k}}(\mathrm{N_{bs}} + 1)$ and sends it to the BS. The BS repeats this operation with its own key and checks whether the result of this operation corresponds to the received MAC. If the MAC is verified correctly, both devices can use *k*, (hereafter denoted as $\mathrm{K_s}$), to securely communicate with each other, otherwise they execute the key establishment protocol again.

Reporting Medical Data to the HO

In this stage the patient's PM sends medical data to the HO. Fig 3 depicts the processes executed by the entities.

PM: The PM performs the following steps.

1. It generates a fresh nonce, $\mathrm{N_{pm}}$, that is used to compute two fresh pseudonyms. First, it computes a pseudonym for itself, i.e. $\mathrm{PS_{pm}} = PRF_{\mathrm{K_{pm\text{-}ps}}}(\mathrm{ID_{pm}} \parallel \mathrm{N_{pm}})$, where *PRF* is a pseudorandom function (e.g. a secure block cipher) and $\mathrm{ID_{pm}}$ is the PM's real identity (e.g. serial number). This pseudonym is only used once (i.e. in the message sent from the PM to the HO). Next, it computes a pseudonym for the HO, i.e. $\mathrm{PS_{ho}} = PRF_{\mathrm{K_{ho\text{-}ps}}}(\mathrm{ID_{ho}} \parallel \mathrm{N_{pm}})$, where $\mathrm{ID_{ho}}$ is the HO's real identity. This pseudonym is used in a pair of messages (i.e. the one sent from the PM to the HO and vice versa). Both pseudonyms protect the patient's privacy while communicating with the HO, i.e. the PM uses $\mathrm{PS_{pm}}$ and $\mathrm{PS_{ho}}$ instead of $\mathrm{ID_{pm}}$ and $\mathrm{ID_{ho}}$.

2. It encrypts the patient's medical data, d, and a counter, $\mathrm{ct_{pm\text{-}ho}}$, using the secret key it shares with the HO, $\mathrm{K_{pm\text{-}ho}}$, i.e. $(\mathrm{C, T})_{\mathrm{pm\text{-}ho}} = AE_{\mathrm{K_{pm\text{-}ho}}}(\mathrm{ct_{pm\text{-}ho}} \parallel \mathrm{d})$. Since PMs do not contain a precise clock, a counter is used to prevent replay attacks. The counter is initialised to zero every time a new session key between the BS and the PM is established. For better performance, the encryption method used is an authenticated encryption scheme (e.g. AES-CCM [133]), which outputs a ciphertext, C, and an authentication tag, T.

3. It constructs a message, $\mathrm{M_{pm\text{-}bs}} = \mathrm{PSs} \parallel (\mathrm{C, T})_{\mathrm{pm\text{-}ho}} \parallel \mathrm{N_{pm}}$, where $\mathrm{PSs} = \mathrm{PS_{pm}} \parallel \mathrm{PS_{ho}}$, and encrypts it using the session key previously established with the BS, $\mathrm{K_s}$, i.e. $(\mathrm{C, T})_{\mathrm{pm\text{-}bs}} = AE_{\mathrm{K_s}}(\mathrm{ct_{pm\text{-}bs}} \parallel \mathrm{M_{pm\text{-}bs}})$.

4. It sends $\mathrm{msg_{pm\text{-}bs}} = (\mathrm{C, T})_{\mathrm{pm\text{-}bs}}$ to the BS.

BS: Upon receiving $\mathrm{msg_{pm\text{-}bs}}$, the BS performs the following steps.

1. It verifies the authenticity of $\mathrm{msg_{pm\text{-}bs}}$ and decrypts the ciphertext to obtain $(\mathrm{ct_{pm\text{-}bs}} \parallel \mathrm{M_{pm\text{-}bs}})$. It then checks whether the counter, $\mathrm{ct_{pm\text{-}bs}}$, is valid, i.e. if

it is higher than the counter of the last received message. If this condition is satisfied, the message is accepted, otherwise it is rejected.

2. It generates a random session ID, S_{id}, that is valid for a pair of messages and allows the HO to anonymously send a message back without knowing which specific BS sent the message. Then it encrypts S_{id} along with $M_{pm\text{-}bs}$ using the public key of the DC, PK_{dc}, i.e. $C_{bs\text{-}dc} = E_{PK_{dc}}(S_{id} \parallel M_{pm\text{-}bs})$.

3. It constructs a message, $M_{bs\text{-}dc} = (TS_{bs} \parallel C_{bs\text{-}dc})$, where TS_{bs} is a timestamp of the BS used to counter replay attacks. TS_{bs} does not need to be kept secret and is sent unencrypted to the DC; however, its integrity is protected by means of a digital signature. This allows the DC to check the freshness of the message before decrypting the message and verifying its authenticity.

4. It generates a signature on $M_{bs\text{-}dc}$ using its private key, $Sig_{SK_{bs}}(M_{bs\text{-}dc})$. Then it constructs a message, $msg_{bs\text{-}dc} = M_{bs\text{-}dc} \parallel Sig_{SK_{bs}}(M_{bs\text{-}dc})$, and sends it to the DC via an anonymous communication channel.

DC: Upon $msg_{bs\text{-}dc}$ reception, the DC performs the following steps.

1. It verifies the freshness of $msg_{bs\text{-}dc}$ by checking TS_{bs} and the authenticity of the message using the BSs group public key, $PK_{bs_{group}}$. It then decrypts $C_{bs\text{-}dc}$ to obtain $(S_{id} \parallel M_{pm\text{-}bs})$, where $M_{pm\text{-}bs} = (PS_{pm} \parallel PS_{ho} \parallel (C, T)_{pm\text{-}ho} \parallel N_{pm})$.

2. It retrieves ID_{ho} by computing $PS'_{ho} = PRF_{K_{ho\text{-}ps}}(ID_{ho_i} \parallel N_{pm})$ for all HOs, $(ID_{ho_1}, \ldots, ID_{ho_n})$ until the DC finds a match, i.e. PS'_{ho} equals PS_{ho}.

3. It constructs a message, $M_{dc} = (PS_{pm} \parallel ID_{ho} \parallel (C, T)_{pm\text{-}ho} \parallel N_{pm})$, in which PS_{ho} is replaced with ID_{ho}. Then it encrypts the message and the session ID, S_{id}, using the public key of the HO, PK_{ho}, i.e. $C_{dc\text{-}ho} = E_{PK_{ho}}(S_{id} \parallel M_{dc})$.

4. It constructs a message, $M_{dc\text{-}ho} = (TS_{dc} \parallel C_{dc\text{-}ho})$, where TS_{dc} is a timestamp of the DC, and generates a signature on $M_{dc\text{-}ho}$ using its private key, $Sig_{SK_{dc}}(M_{dc\text{-}ho})$. Finally, it appends the signature to $M_{dc\text{-}ho}$ in order to form a message, $msg_{dc\text{-}ho} = M_{dc\text{-}ho} \parallel Sig_{SK_{dc}}(M_{dc\text{-}ho})$, and sends it to the HO.

HO: Upon $msg_{dc\text{-}ho}$ reception, the HO performs the following steps.

1. It verifies the freshness of $msg_{dc\text{-}ho}$ by checking TS_{dc} and the authenticity of the message by checking the validity of $Sig_{SK_{dc}}(M_{dc\text{-}ho})$. It then decrypts $C_{dc\text{-}ho}$ to obtain $(S_{id} \parallel M_{dc})$, where $M_{dc} = (PS_{pm} \parallel ID_{ho} \parallel (C, T)_{pm\text{-}ho} \parallel N_{pm})$.

2. It retrieves ID_{pm} by computing $PS'_{pm} = PRF_{K_{pm\text{-}ps}}(ID_{pm_i} \parallel N_{pm})$ for all PMs, $(ID_{pm_1}, \ldots, ID_{pm_w})$, where w is the number of patients with a PM registered in the HO, until a match is found, i.e. PS'_{pm} equals PS_{pm}.

3. It searches for ID_{pm} in its database to retrieve $K_{pm\text{-}ho}$. Using this key, the

HO verifies and decrypts $(C, T)_{pm\text{-}ho}$ to obtain $(ct_{pm\text{-}ho} \parallel d)$. Next, the HO verifies the freshness of the message by checking if the counter, $ct_{pm\text{-}ho}$, is higher than the counter of the previously received message. Only if this condition is fulfilled, the HO accepts the patient's medical data, d, as authentic and genuine.

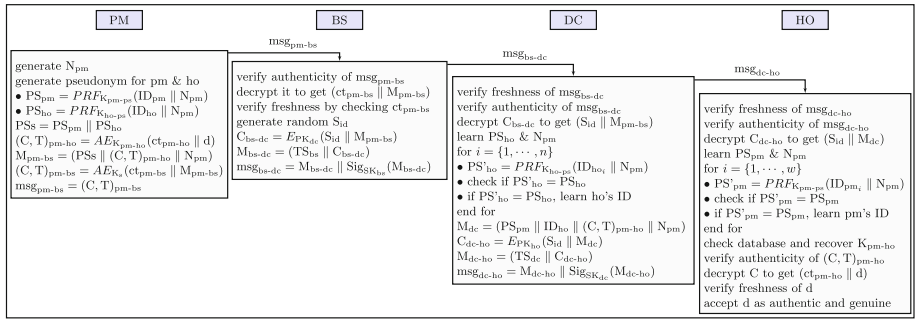

Figure 3: Medical data reporting protocol.

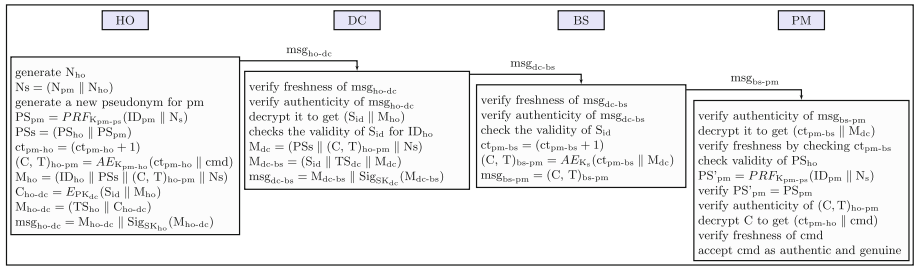

Figure 4: PM's reprogramming protocol data reporting protocol.

PM Reprogramming Stage After examining the patient's medical data, the doctor can send the necessary command(s) to adjust the PM's settings. This stage, which can only take place if the doctor is on-line, is described next and shown in Fig. 4.

HO: The HO performs the following steps.

1. It generates a fresh nonce, N_{ho}, to create a fresh pseudonym for the PM, PS_{pm}. Then it increases the counter $ct_{pm\text{-}ho}$, and encrypts it along with the command, cmd, using $K_{pm\text{-}ho}$, i.e. $(C, T)_{ho\text{-}pm} = AE_{K_{pm\text{-}ho}}(ct_{pm\text{-}ho} \parallel cmd)$.

2. It constructs a message, $M_{ho} = (ID_{ho} \parallel PSs \parallel (C, T)_{ho\text{-}pm} \parallel Ns)$, where $PSs = (PS_{pm} \parallel PS_{ho})$ and $Ns = (N_{pm} \parallel N_{ho})$. Next it encrypts M_{ho} and the session ID using the public key of the DC, i.e. $C_{ho\text{-}dc} = E_{PK_{dc}}(S_{id} \parallel M_{ho})$.

3. It constructs a message $M_{ho\text{-}dc} = (TS_{ho} \parallel C_{ho\text{-}dc})$ and generates a signature on it, $Sig_{SK_{ho}}(M_{ho\text{-}dc})$. Then it constructs a message, $msg_{ho\text{-}dc} = M_{ho\text{-}dc} \parallel Sig_{SK_{ho}}(M_{ho\text{-}dc})$, and sends it to the DC.

DC: Upon $msg_{ho\text{-}dc}$ reception, the DC performs the following.

1. It verifies the freshness and authenticity of $msg_{ho\text{-}dc}$, and decrypts it to obtain $(S_{id} \parallel M_{ho})$. Once it learns ID_{ho} and S_{id}, it checks if S_{id} is a valid session ID for this HO, i.e. if a message containing this session ID was previously sent to this specific HO. It then constructs $M_{dc\text{-}bs} = (S_{id} \parallel TS_{dc} \parallel M_{dc})$, where $M_{dc} = (PSs \parallel (C, T)_{ho\text{-}pm} \parallel Ns)$, generates a signature on it, $Sig_{SK_{dc}}(M_{dc\text{-}bs})$, and constructs a message, $msg_{dc\text{-}bs} = M_{dc\text{-}bs} \parallel Sig_{SK_{dc}}(M_{dc\text{-}bs})$. Finally, $msg_{dc\text{-}bs}$ is sent to the BS over the anonymous communication channel previously used.

BS: Upon $msg_{dc\text{-}bs}$ reception, the BS performs the following.

1. It verifies the freshness and authenticity of $msg_{dc\text{-}bs}$ before checking the validity of S_{id}, i.e. checking if this session ID is the same as one previously generated by the BS. It then increases the counter by one, i.e. $ct_{pm\text{-}bs} = (ct_{pm\text{-}bs} + 1)$. It encrypts $ct_{pm\text{-}bs}$ and M_{dc} using the session key, i.e. $(C, T)_{bs\text{-}pm} = AE_{K_s}(ct_{pm\text{-}bs} \parallel M_{dc})$, and constructs and sends a message, $msg_{bs\text{-}pm} = (C, T)_{bs\text{-}pm}$, to the PM.

PM: Upon $msg_{bs\text{-}pm}$ reception, the PM performs the following steps.

1. It verifies the authenticity of $msg_{bs\text{-}pm}$, decrypts it to obtain $(ct_{pm\text{-}bs} \parallel M_{dc})$ and verifies the freshness of M_{dc} by checking $ct_{pm\text{-}bs}$. It then verifies the validity of PS_{ho}, i.e. checks if PS_{ho} has been previously generated at the PM.

2. It uses $K_{pm\text{-}ps}$, its own ID and N_s to verify the PM's pseudonym, $PS'_{pm} = PRF_{K_{pm\text{-}ps}}(ID_{pm} \parallel N_s)$.

3. It verifies the authenticity of $(C, T)_{ho\text{-}pm}$, decrypts it to obtain $(ct_{pm\text{-}ho} \parallel cmd)$ and verifies the freshness of cmd. If the verifications are correct, it accepts cmd as an authentic and genuine command sent from the patient's HO.

5 Security Analysis

Message Authenticity: Messages exchanged between any of the entities contain either a digital signature of the message originator or a MAC. Assuming

that a standard digital signature scheme (e.g. RSA or Schnorr variant of ECDSA [79]) or a MAC algorithm (e.g. AES CBC-MAC) are used, our protocol guaranties source authentication, message integrity and non-repudiation (only with digital signatures). Thus, attacks that attempt to modify the messages in transit can be detected. All messages include either a counter or a timestamp to ensure freshness, and hence prevent replay attacks (satisfy (S1) and (S2)).

Confidentiality of Medical Data and Commands: Medical data and commands to modify the PM's settings are always encrypted twice (i.e. in two encryption layers). The inner-layer encryption is carried out between the PM and the HO. This provides end-to-end security. The outer-layer encryption is performed between any two communicating entities, and used to hide the pseudonyms from adversaries. In addition, it helps to prevent some types of denial-of-service attacks (satisfy (S4)). Assuming that a standard encryption scheme (e.g. AES) is used, it will be hard for eavesdroppers to learn the content of the messages. Only authorised medical staff will be able to access the medical data or send commands (satisfy (S3)).

Patient's Identity Privacy: Each message exchanged between the PM and the HO contains a distinct pseudonym to hide the PM's real ID. This pseudonym is generated using a PRF, e.g. AES-128, and the symmetric key that is known only to the PM and the authorized HO. AES-128 can be used as a PRF as long as the number of messages for one key is less than 2^{40}, which corresponds to more than 4,000 encrypted messages per second exchanged between the HO and the PM (assuming that the battery lasts 7 years). All unauthorized internal entities (i.e. BSs and DC) as well as external adversaries will not be able to obtain the PM's real ID. Only the authorized hospital can recover the real ID of the PM, and link the medical data to a specific patient (satisfy (P1.1)).

Hospital's Identity Privacy: For each pair of messages exchanged between the PM and the HO, the PM generates a distinct pseudonym to hide the real ID of the hospital where it sends medical data. This pseudonym is generated using a PRF, e.g. AES-128, and the symmetric key shared between all PMs and the DC. As explained above, AES-128 is a secure PRF if the number of encrypted messages is less than 2^{40}. However, external adversaries cannot have access to the pseudonyms produced by the PRF, since pseudonyms are always sent in an encrypted format between the communicating entities (satisfy (P1.2)).

Location Privacy: A low-latency anonymous channel (e.g. a Mix network) between the BSs and the DC in combination with a group signature scheme prevents the DC from learning the location of the BS while being used by the patient (satisfy (P1.3)).

Session Unlinkability: Fresh and random pseudonyms are used in each message exchanged between the PM and the HO. Therefore, by looking at the exchanged messages, no unauthorized entity can link different sessions or learn if two messages have been sent by the same PM (satisfy (P1.4)).

Protection against Stolen BSs or Pre-owned PMs: Since BSs are simply relay devices that do not have any pre-installed shared secrets with PMs, adversaries who get a BS cannot access the content (i.e. medical data and commands) of the messages exchanged between the PM and the HO. In addition, adversaries who obtain a new or a pre-owned PM cannot send data to the HO. Upon a PM replacement, the old PM's ID and the corresponding keys are removed from the database so that the patient is no longer linked to the old PM.

6 Conclusions

In this paper, we proposed a protocol that provides end-to-end security between a PM and a HO while preserving the patient's privacy. Each PM uses two fresh pseudonyms for hiding the unique PM's ID and the HO where the medical data is sent. These pseudonyms allow the DC to forward the medical data to the authorized HO without learning to whom the data belongs to, and prevent adversaries from discovering the PM's real ID and to which hospital the medical data is sent. In addition, all BSs sign their messages using a group signature scheme and send them to the DC over an anonymous channel. This allows (i) the DC to verify the authenticity of the messages and (ii) the HO to link the medical data to the patient without learning which specific BS sent the messages (i.e. the location of the patient). Moreover, we presented an IPI-based key establishment protocol that allows a PM and a BS to agree on a symmetric key without using public-key cryptography or any pre-installed shared secrets.

7 Acknowledgements

The authors would like to thank George Petrides and the anonymous reviewers for their helpful comments. This work was supported by KIC InnoEnergy SE via KIC innovation project SAGA, and the Research Council KU Leuven: C16/15/058.

Publication

On the Feasibility of Cryptography for a Wireless Insulin Pump System

Publication Data

Eduard Marin, Dave Singelée, Bohan Yang, Ingrid Verbauwhede, and Bart Preneel, "On the Feasibility of Cryptography for a Wireless Insulin Pump System," In Proceedings of the 6th Conference on Data and Application Security and Privacy (CODASPY 2016), 8 pages, 2016.

Contributions

- Principal author, except for the MSP430 implementation.

On the Feasibility of Cryptography for a Wireless Insulin Pump System

Eduard Marin, Dave Singelée, Bohan Yang, Ingrid Verbauwhede, and Bart Preneel

ESAT-COSIC and iMinds, KU Leuven, Belgium

Abstract. This paper analyses the security and privacy properties of a widely used insulin pump and its peripherals. We eavesdrop the wireless channel using Commercial Off-The-Shelf (COTS) software-based radios to intercept the messages sent between these devices; fully reverse-engineer the wireless communication protocol using a black-box approach; and document the message format and the protocol state-machine in use. The upshot is that no standard cryptographic mechanisms are applied and hence the system is shown to be completely vulnerable to replay and message injection attacks. Furthermore, sensitive patient health-related information is sent unencrypted over the wireless channel.

Motivated by the results of our attacks, we study the feasibility of applying cryptography to protect the data transmitted over the air and prevent unauthorized access to the insulin pump. We present a solution based on AES in combination with an updated message format optimized for energy consumption. We implement our solution on a 16-bit micro-controller and evaluate its security properties and energy requirements. Finally, we discuss potential strategies for further reducing the energy consumption.

1 Introduction

Wearable and Implantable Medical Devices (IMDs) such as pacemakers, neurostimulators and insulin pumps are currently used to monitor and treat physiological conditions within the patient's body. For example, patients with diabetes require a precise daily dosage of insulin, combined with a strict schedule for diet, physical activity and insulin injections. The dosage can be automated via an insulin pump. Unlike self-injection, insulin pumps provide more flexibility, tighter control of the patient's diabetes and better predictability by delivering more precise insulin doses.

Nowadays, most insulin pumps have a radio that enables wireless communication with various peripherals (e.g. a glucose meter), a more powerful embedded processor and connectivity to a back-end computing infrastructure. While these advances bring substantial clinical benefits, new security and privacy threats emerge. More specifically, the wireless interface of these devices is of particular concern as it could be exploited by adversaries to perform remote attacks; adversaries might eavesdrop the wireless channel to compromise the patient's privacy, or even worse, send unauthorized commands to the insulin pump.

Contributions. Our first contribution is to show that (at least) two commercial insulin pump models use proprietary wireless communication protocols without any security protection. By fully reverse-engineering the wireless communication protocol, we are able to: (i) discover sensitive information on the patient sent in the clear over the air and (ii) send unauthorized commands to the insulin pump. Our second contribution is a practical case study on the feasibility of using cryptography to secure the wireless channel between these devices. We present a cryptographic AES-based solution with an updated message format optimized for energy consumption, an important requirement for these devices. We evaluate the security properties and energy requirements of our solution. Finally, we implement our solution on a 16-bit micro-controller, similar to the one used in the insulin pump system, and discuss possible ways to further reduce the energy consumption.

Disclosure of results. Our study examines a widely adopted insulin pump system that is being used by many diabetic patients worldwide. Given the sensitive nature of our results, we deliberately omit some details (e.g. model and manufacturer) as they would allow anyone to easily reproduce our attacks on insulin pump systems used by patients.

Paper outline. The remainder of this paper is organized as follows. Section 2 provides an overview of related work. Section 3 describes the devices that are part of the insulin pump system. Section 4 shows the laboratory setup used to eavesdrop the wireless channel and perform the remote attacks. The methodology of how to reverse-engineer the proprietary wireless communication protocol is explained in Section 5. Section 6 shows several software radio-based attacks that are carried out on the insulin pump, whereas a cryptographic solution to secure the wireless link is presented and evaluated in Section 7, particularly focusing on the energy consumption. Section 7 provides a final conclusion on the paper.

2 Related Work

2.1 Attacks on Medical Devices

Recently, it has been shown that some medical devices are vulnerable to security attacks. Halperin et al. surveyed a wide range of security issues, and explained the need to balance the security and privacy of IMDs with safety and effectiveness [69]. Halperin et al. also performed a security and privacy analysis of a widely used Implantable Cardioverter Defibrillator (ICD). More specifically, they carried out several software radio-based attacks that could compromise both the safety and privacy of the patient [70]. Similar attacks were also executed on the wireless channel between an insulin pump and a remote control, as shown by Li et al. [86]. The main focus of their paper was to show different types of attacks that adversaries could perform after fully reverse-engineering the wireless communication protocol. In this paper, we validate and extend their attacks by fully reverse-engineering the wireless communication protocol between all the peripherals of the insulin pump system. This includes the wireless link between: (i) the remote control and the insulin pump, (ii) the glucose meter and the insulin pump, (iii) the glucose sensor and the insulin pump and (iv) the USB stick and the insulin pump. This paper also describes the methodology of how to reverse-engineer the proprietary wireless communication protocol.

2.2 Countermeasures

Researchers have proposed several countermeasures to solve or mitigate the vulnerabilities found on medical devices. Li et al. suggested using rolling codes, as used in garage doors, to protect against unauthorized entities [86]. However, some rolling codes do not offer strong authentication. Another weak point of their proposal is that it assumes that the remote control and the insulin pump share an encryption key, but it does not explain how this key is updated and revoked. The solution proposed by Gollakota et al. consists of introducing an external device, known as “shield”, which acts as a relay between the IMD and the external device programmer [67]. The “shield” has been prototyped for a cardiac device to mitigate some of the security problems, but it has not been tested for wearable medical devices, such as insulin pumps. Anomalous detection is also a promising technique that could be used in combination with other security mechanisms to prevent unauthorized access, as shown by Hei et al. [72]. They proposed to use past glucose trends to detect anomalous behaviours. However, while strict policies can increase the detection rate, they may also result in false alarms. Undoubtedly, cryptography is the only strong approach for

securing the data transmitted over the wireless communication and preventing unauthorized access. The use of cryptography, though, is challenging because IMDs are resource-constrained devices that require reduced size, low peak power and a low duty cycle. Key management also presents important challenges in terms of scalability, usability and the capacity to deal with emergency situations [114]. This makes it non-trivial to propose cryptographic solutions that increase the level of security and privacy of the system without jeopardizing the patient's safety. In this paper, we address this problem by proposing an AES-based solution optimized for the insulin pump system. Furthermore, we investigate how the energy cost of this security solution can be reduced without compromising the security.

3 Insulin Pump System

This section describes the devices that are part of the insulin pump system (see also Figure. 1).

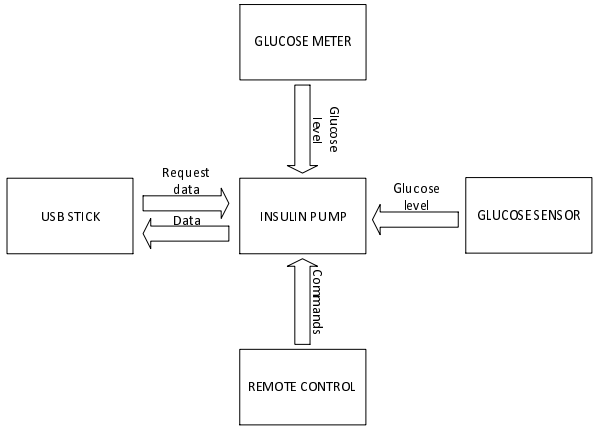

Figure 1: Insulin pump system.

The *insulin pump* can administer two types of insulin doses: a bolus dose, which is pumped quickly to the patient after meals, or a basal dose, which is continuously delivered at an adjustable rate. The *remote control* allows patients to send commands wirelessly to their insulin pump from a distance up to 1-2 meters. The *glucose meter* and the *glucose sensor* (attached to the body) measure the glucose's concentration in the patient's blood, and send this value wirelessly to the insulin pump. This is then used as an indicator for the patient to adjust the dose. The *USB stick (with software for therapy management)* is a

remote monitoring tool that collects data from the patient's insulin pump, and then uploads these data to a website managed by the manufacturer.

Before any of these devices can communicate with the insulin pump, the patient needs to manually enter the device's serial number on the insulin pump (except for the USB stick). This serial number is typically printed on the back of the device. This pairing process only needs to be done once. From then onwards, these devices can interact with the insulin pump as long as the patient does not modify the list of valid serial numbers stored on the insulin pump. If any of these devices is lost, damaged or stolen, the patient can manually remove its serial number from the list of valid devices, and enter the serial number of the new device.

4 Laboratory Setup

In our laboratory we employed available Commercial Off-The-Shelf (COTS) hardware including a Universal Serial Radio Peripheral (USRP) [12], an oscilloscope, an insulin pump, a remote control, a glucose meter, a glucose sensor and a USB stick. Initially, we used the USRP, a commercial antenna and a receiver program to eavesdrop the wireless channel and intercept messages. Using this setup, the sending/receiving communication range is more than 5 meters. In practice, adversaries can extend this range to several orders of magnitude. At a later stage, we were able to transmit messages by means of the USRP, the antenna and a transmitter program. Our receiver and transmitter programs were built in LabVIEW [7], a software tool which allows to interact with the USRP.

5 Methodology

The first step in the security and privacy analysis of this insulin pump system is to understand which data are sent and how these data are exchanged between devices. This is a difficult task because the protocol specifications are kept secret, since IMDs manufacturers often rely on obscurity to provide security, also known as *security-through-obscurity*. However, experience has shown that these proprietary solutions can be broken via several reverse-engineering techniques. More specifically, we follow a black-box based approach, i.e. we only give some inputs to the devices and then look at their outputs. In other words, we conduct an operation on the device (e.g. we press any of the remote control buttons), and then we look at the produced message format.

Our black-box approach can be divided into four steps:

1. Find the wireless communication parameters: Several wireless communication parameters, including the transmission frequency, modulation and symbol rate, need to be first discovered. This step is crucial, as using slightly different values would result in an erroneous received message.

2. Reverse-engineering: Secondly, we eavesdrop the wireless channel to intercept messages sent from any of the peripherals to the insulin pump (and vice versa). We then analyse these messages to discover both the message format as well as the protocol state-machine. For example, the existence of the remote control's serial number within the message might be revealed by intercepting and comparing two messages sent by two different remote controls while performing the same action, i.e. pressing the same button.

3. Obtain the serial number: Most active attacks can only be mounted if the serial number of a valid device, i.e. in the list of patient's devices with which the insulin pump can communicate, is known. There are several means to obtain the serial number of any of the patient's devices: (i) we could either peek at the back of the device itself, (ii) get it through an insider working in the hospital, (iii) eavesdrop once the wireless channel or (iv) brute-force it.

4. Perform the attacks: Once the previous steps are completed, various attacks including replay, message injection and privacy attacks can be carried out on the insulin pump. In the next section, we will describe these attacks more in detail.

6 Experimental Results

This section describes various types of active and passive software radio-based attacks that we can perform on the insulin pump.

6.1 Wireless Communication Parameters

The *transmission frequency* of any wireless device is publicly available on-line, and it can be easily obtained through its Federal Communications Commission (FCC) ID [5]. By checking the remote control's FCC ID, we discovered that this device transmits at 868.35 MHz. Fig. 2 shows the waveform of the signals transmitted by the remote control. By observing this waveform, we found that the *modulation scheme* uses the presence of a carrier wave to indicate a binary '1' and its absence to indicate a binary '0'. This type of modulation is known as

On-Off Keying (OOK). To find the *symbol rate*, we opened our remote control and looked at the raw bits (i.e. bits before modulation) to be transmitted by tapping one pin of its micro-controller. Using the oscilloscope, we measured the duration of a bit, which corresponds to the duration of a symbol. Following the same steps, we found that the other peripherals also use the same wireless communication parameters.

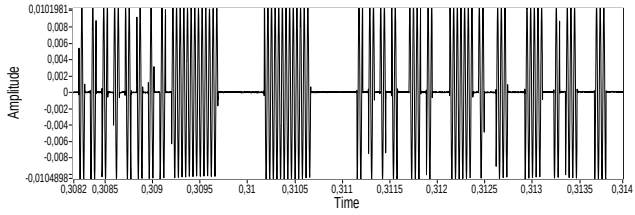

Figure 2: Waveform of the signal transmitted by the remote control.

6.2 Reverse-engineering the Wireless Communication Protocol

In this section, we show how to reverse-engineer the wireless communication protocol between the insulin pump and its peripherals.

We found that the messages sent from any of the peripherals to the insulin pump (and vice versa) consist of common message fields and information bits that depend on the peripheral being used. In particular, all messages have a common start-of-frame and frame synchronization sequences. The former consists of a series of "1s" and "0s" sent consecutively to wake up the insulin pump from power saving mode, whereas the latter is used to indicate that the information bits are about to start. Next, there is a 6-bit sequence used to distinguish between the four types of devices that can communicate with the insulin pump. We will now explain how to discover the information bits sent by each peripheral more in detail.

Remote control - insulin pump

As explained in Section 2, in 2011, Li et al. reverse-engineered the wireless communication protocol between the remote control and the insulin pump [86]. In our experiments, presented below, we obtained similar results and hence validated their findings.

We started our tests by capturing the messages sent by the remote control while performing the same operation (i.e. pressing the same remote control's button)

several times. The result of this test showed that the remote control produces messages that differ in 24 bits each time. On the one hand, by computing the entropy of these bits, we determined that 12 bits seemed to be uniformly distributed, which made us think that a Cyclic Redundancy Check (CRC) could be used to detect bit errors in the transmissions. CRCs are easy to implement and very effective at detecting bit errors caused by noise in the communication channel. In order to find the CRC generator polynomial, rather than bruteforcing all possible polynomials, we took the GCD of two observed messages (in polynomial form). This allowed us to discover that a standard CRC commonly used in mobile networks, known as CRC-8-WCDMA, is being used. On the other hand, the remaining 12 bits are repeated after 256 messages and hence they are used as a message counter, which is increased every time the patient presses any of the remote control's buttons.

Afterwards, we ascertained whether pressing different buttons on the remote control or using different remote controls causes some part(s) of the message to change. These tests led us to conclude that a 12-bit sequence depends on the button being pressed, and a 36-bit sequence depends on the remote control being used. Furthermore, by looking at two 36-bit sequences and their corresponding 6-digit remote control's serial numbers, we found that each digit is represented by 6 bits. A mapping sequence is then used to convert each 6-bit sequence to a 4-bit hexadecimal number. Thus, due to this mapping sequence, the serial number only contains 24 unique bits. Note that for security reasons we do not disclose the table containing the mapping sequences in the paper. Fig. 3 shows the remote control's message format.

Figure 3: Remote control's message format.

Glucose meter - insulin pump

Several messages sent by two different glucose meters, wherein a wide range of glucose levels were measured, were first intercepted. Based on our previous results, we investigated whether using different glucose meters causes some part(s) of the message to change. We found that the glucose meter's serial number is represented by 36 bits (6-bit per digit) and that it uses the mapping sequence previously found. This is followed by the information field, which remains empty (i.e. all zeros) for all captured messages. Subsequently, we grouped messages with identical and different glucose level, separately, in

several clusters. By comparing the messages in these clusters, we discovered that 12 bits are used to transmit the measured glucose level. Finally, there is a 12-bit CRC that is computed using the CRC-8-WCDMA. Fig. 4 shows the glucose meter's message format.

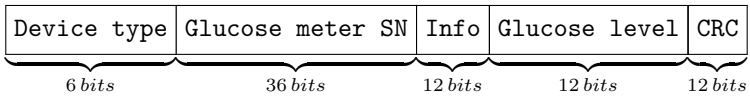

Figure 4: Glucose meter's message format.

Glucose sensor - insulin pump

Given the similarities between the previous message's formats, the first step was to determine whether the messages sent by the glucose sensor contain its serial number. This test showed that, unlike the previous cases in which we found a 36-bit serial number, the glucose sensor's serial number is represented by 84 bits. In order to discover the rest of the message, we then looked at several consecutive messages sent by the glucose sensor. This allowed us to find a 6-bit counter along with the current and past eight measured glucose levels. Because of the longer message's length in comparison with the messages sent by the previous peripherals, these messages use a 24-bit CRC (instead of a 12-bit CRC). We found that the CRC being used is the standard CRC-16-CCITT. Fig. 5 shows the glucose sensor's message format.

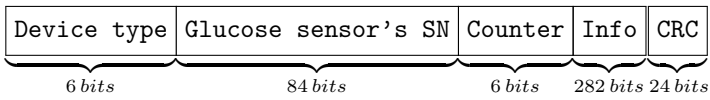

Figure 5: Glucose sensor's message format.

USB stick - insulin pump

Without looking at the bits being transmitted, we first studied the message exchange process between the USB stick and the insulin pump. For this, we placed the insulin pump close to our USRP while keeping the USB stick (connected to our laptop) further away, thus, getting more power from the insulin pump, as shown in Figure. 6. During a communication session, the USB stick and the insulin pump send to each other a fix number of messages. The USB stick always initiates the communication by sending a "wake up" message several times to the insulin pump. If the remote control and the insulin pump

are within the same communication range, the latter replies to this message and the interrogation process starts, as shown in Figure. 6. From that point onwards, two different consecutive phases can be distinguished: in the first phase, the USB stick requests data such as the model, firmware or the current settings from the insulin pump, whereas in the second phase the insulin pump first sends a message to the USB stick and the latter responds with an ACKnowledgement message (ACK).

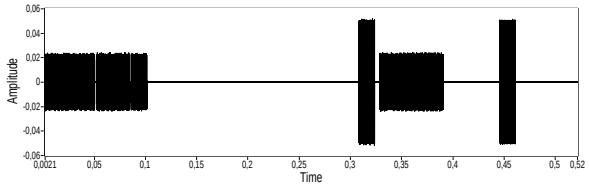

Figure 6: Communication between the USB stick (connected to the computer) and the insulin pump. From left to right, the message sent several times by the USB stick to wake up the insulin pump, the response sent by the insulin pump to the first USB stick's message, the second message sent by the USB stick to the insulin pump and the response sent by the insulin pump to the second USB stick's message.

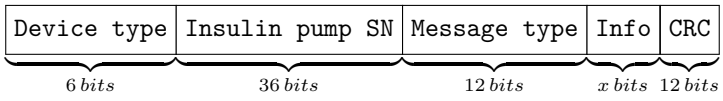

Figure 7: USB stick/insulin pump's message format.

Once we understood the message exchange process between these devices, we started to analyse the messages sent from the USB device to the insulin pump (and vice versa). A first test consisted on capturing several messages sent by the USB stick while interrogating two different insulin pumps. This test revealed that the insulin pump's serial number is transmitted in both messages; the ones sent by the USB stick and the ones sent by the insulin pump. We found that the mapping sequence previously discovered is also used. By comparing two consecutive messages sent by the USB stick and the insulin pump, respectively, we found a 12-bit message type (or message identifier) sequence which indicates the USB stick message the insulin pump replies to.

Subsequently, insulin pump's data such as the model, firmware or current settings are transmitted within the message. The length of this field (denoted by x in Figure. 7) depends on the information requested/provided by the devices.

Finally, similarly to the cases above, there is a 12-bit CRC at the end of the messages that is computed using the CRC-8-WCDMA. Fig. 7 shows the message format being used by the USB stick and the insulin pump.

6.3 Software Radio-based Attacks

In this section, we focus on the active and passive attacks that we can carry out on the insulin pump after having reverse-engineered the wireless communication protocol.

Replay attacks: We first investigated the feasibility of carrying out replay attacks, as these attacks can even be conducted without reverse-engineering the protocol; knowing the transmission frequency being used by the devices is sufficient. At first, we intercepted several messages sent by the remote control to the insulin pump and simply re-sent each message without demodulating the signal. Our experiments demonstrated that some sort of anti-replay mechanism is being used. This anti-replay mechanism does not allow to re-send the last message accepted by the insulin pump, but it does not impede us to send any other arbitrary message previously recorded. Thus, just by obtaining two different messages while performing an action (e.g. activate the insulin pump), we can alternatively re-send these messages as many times as needed in a protocol instance. Following the same approach, we also successfully performed this attack on the wireless link between: (i) the glucose meter and the insulin pump and (ii) the USB stick and the insulin pump. In both cases, messages were accepted the first time they were re-sent, meaning that no anti-replay mechanism is being used.

Message injection attacks: With the knowledge gained during the reverse-engineering process, we can create messages containing a valid serial number and successfully send them to the insulin pump. As a result, we can activate/suspend the insulin pump, deliver a high amount of insulin to the patient and send any arbitrary glucose value to the insulin pump. In addition to this, two different types of message injection attacks can be mounted on the wireless link between the USB stick and the insulin pump. On the one hand, we can emulate the insulin pump's behaviour by replying to USB stick transmissions. This attack, though, has several limitations. Since the USB stick initiates the communication, triggered when a doctor clicks a button in the website managed by the manufacturer, the attacker has to wait for such an event and hijack the session, potentially even having to block the communication from the genuine insulin pump. On the other hand, we can also impersonate the USB stick to trigger an interrogation process. This could compromise both the security

and privacy of the patient, as it might be launched either to discover sensitive patient health-related information or to drain the insulin pump's battery.

Privacy attacks: Our analysis of the wireless communication protocol reveals that messages sent from any of the peripherals to the insulin pump (and vice versa) are not encrypted. Therefore, these messages disclose the type of device with which the insulin pump is communicating and its serial number, the current and past patient glucose levels, the insulin pump's model/software version, the current insulin pump's settings, the basal rate information or the total amount of insulin taken by the patient.

7 Security Defences

In the previous sections, we executed several active and passive attacks on the insulin pump that could compromise the safety and privacy of the patient. In this section, we present a cryptographic solution based on AES [52] that provides data confidentiality, authentication and freshness at a reasonable energy cost. Furthermore, we propose an optimized message format to reduce the energy cost. Without loss of generality, we will only focus on the wireless link between the remote control and the insulin pump. However, our solution can be applied as well to the wireless link between the insulin pump and any of the other peripherals.

7.1 Key Management

The remote control and the insulin pump need to share two independent symmetric cryptographic keys: one for encryption and one for authentication. We will now briefly explain the process of generating, transporting, updating and revoking these keys.

Key generation: To reduce the software footprint and the redundant computation on the remote control, both keys are generated by the insulin pump. To guarantee an appropriate level of entropy, a true randomness source is required. This randomness could originate either from the Static Random-Access Memory (SRAM) [130] or a True Random Number Generator (TRNG) [15].

Key transport: After the key generation process, both keys have to be securely transported from the insulin pump to the remote control. For this, we suggest to make physical contact between both devices via a secure channel (e.g. we connect a cable between devices), similarly as the Resurrecting Duckling approach proposed by Stajano and Anderson [127].

Key update: The patient is required to update the cryptographic keys every time the battery is replaced, i.e. every three weeks. During this process, the anti-replay counter is re-initialised to zero.

Key revocation: If the patient's remote control is lost, stolen or damaged, both keys have to be revoked. If this occurs, the key generation procedure has to be executed with the new remote control. From that point onwards, the old keys are removed from the insulin pump.

7.2 Our Approach

AES-based encryption and MAC

Our solution makes use of two cryptographic primitives: AES-128 in counter (CTR) mode to encrypt messages and AES-128 as a MAC. To find the energy requirements of our solution, we implemented both AES-128 in CTR mode and AES-128 MAC for openMSP430 [13] on a Spartan-6 FPGA. This is a representative mid-range micro-controller similar to the one used in the insulin pump. This software implementation can be easily converted to any other 16-bit micro-controller. Our implementation has been compiled with GNU Tools for Texas Instruments MSP430 micro-controllers optimized for code size with -Os gcc flags. In the rest of this section, we will investigate how to optimize this solution.

Table 1: Implementation on MSP430 @16 MHz, 1.8 V.

Operation	ROM (Byte)	Cycles	Time (μs)	Energy (μJ)
MAC generation	2684	9430	590	2.14
MAC verification	2760	9561	598	2.16
Encryption/Decryption	2664	9404	588	2.13
Encryption + MAC generation	2879	18865	1180	4.27
Decryption + MAC verification	2847	18964	1186	4.30

[1] Original remote control's message format without the mapping sequence.

Table 2: Energy cost (per day) of each solution for the remote control (RC) and the insulin pump (IP).

Solution	Confidentiality	Authentication	Computation cost	Communication cost	Total cost	Cost increase
No security (original system)	✗	✗	RC: 0 mJ IP: 0 mJ	RC: 26.32 mJ IP: 8.77 mJ	RC: 26.32 mJ IP: 8.77 mJ	RC: - IP: -
No security (new message format[1])	✗	✗	RC: 0 mJ IP: 0 mJ	RC: 18.9 mJ IP: 6.30 mJ	RC: 18.9 mJ IP: 6.30 mJ	RC: -28.19% IP: -28.16%
MAC + opt SN encryption	✓	✓	RC: 0.64 mJ IP: 0.65 mJ	RC: 35.10 mJ IP: 11.86 mJ	RC: 35.74 mJ IP: 12.51 mJ	RC: +35.79% IP: +42.64%
Only encryption	✓	✗	RC: 0.32 mJ IP: 0.32 mJ	RC: 20.25 mJ IP: 6.75 mJ	RC: 20.57 mJ IP: 7.07 mJ	RC: -21.84% IP: -19.38%
Only MAC	✗	✓	RC: 0.32 mJ IP: 0.32 mJ	RC: 39.15 mJ IP: 13.05 mJ	RC: 39.47 mJ IP: 13.37 mJ	RC: +50% IP: +52.45%

Communication cost vs. computation cost

Although our solution provides data confidentiality and authentication to prevent the remote attacks previously shown, the use of cryptography increases the energy consumption in both the remote control and the insulin pump. The energy consumption can be divided into two main components: the computation and the communication cost. The former indicates the cost of performing operations such as encrypting/decrypting messages, whereas the latter refers to the cost of transmitting/receiving bits to/from a device.

On the one hand, to calculate the *computation cost*, we measure the number of clock cycles needed to perform an operation. By looking at the power consumption (in the micro-controller specifications [10]) and knowing the number of clock cycles required, we can compute its energy cost, as shown in Table 1. (Note that all the operations described in this table consider a 128-bit block). On the other hand, to compute the *communication cost*, we looked at the datasheet specifications of the remote control's radio transmitter and the insulin pump's radio receiver [16]. As a result, we found that the cost of transmitting and receiving one bit at 868.35 MHz using an OOK modulation scheme is $2.25\mu J$ and $0.75\mu J$, respectively.

Assumptions: To demonstrate the practicality of our solution, we consider the following worst case scenario. We assume that there is a patient in a very crowded hospital full of patients who all use an insulin pump. We assume that each patient has 5 meals per day, during which any of the remote control's buttons are pressed 30 times, so 150 remote control's buttons are pressed per patient daily. Given the 2 meter communication range between the remote control and the insulin pump, we can estimate the interference, i.e. the number of messages received from other patients' remote control. Assuming a density of 4 people per square meter, the number of patients within a radius of two meters is 52. In other words, the insulin pump of the patient receives 7800 messages sent by other patients' remote control daily.

New message format

When observing the original message format (see Figure. 3), we note that several message fields can be optimized or removed when adding security countermeasures. In the new message format, the mapping sequence used to convert each 4-bit hexadecimal sequence to a 6-bit sequence can be removed, as it provides no extra security and it is costly to transmit extra bits. As a result, the length of several fields is reduced, resulting in an energy reduction (see Table 2). In the rest of this section, we will further optimize this new message format to reduce the communication cost without significantly increasing the computation cost.

MAC + optimized SN + encryption

Our solution consists of encrypting the messages sent by the remote control using AES-128 in CTR mode, and then append a AES-128 MAC tag to each message. Intuitively, another possible solution would be to use authenticated encryption like Counter with CBC-MAC (CCM). However, from an energy point of view, there is no advantage on using CCM in comparison with our solution because of the small message length.

Message format: we notice that we can further optimize the message format previously proposed. For example, the CRC used in the original message format to verify the message's integrity can be removed, as the MAC by itself already provides integrity protection. To prevent replay attacks during the three weeks lifetime of the key, a counter needs to be increased every time the patient presses any of the remote control buttons. This counter needs to be large enough to avoid reuse of previous values. In contrast to the original design, the insulin pump will now only accept a message if and only if the counter of the message is higher than the value of the previously received message.

Remote control's serial number optimization: Since the MAC by itself is used to authenticate the remote control, another possible optimization is to reduce the length of the remote control's serial number. However, there is an important trade-off between the communication cost and the computation cost. On the one hand, if there is no remote control's serial number in the messages, the insulin pump would have to verify the MAC of the messages sent by: (i) the patient's remote control and (ii) other patients' remote control within the communication range. This approach increases the computation cost while reducing the communication cost. On the other hand, if the entire remote control's serial number is sent, the insulin pump would only need to verify the MAC of the messages sent by the patient's remote control. However, although this approach requires less computations, more bits need to be sent/received.

Therefore, we investigate if it is possible to reduce the remote control's serial number length while keeping the cost of verifying some extra MACs at a reasonable level. The relation between the energy consumption and the remote control's serial number length for both the remote control and the insulin pump is shown in Figure. 8. When analysing this figure, we notice that the most optimal serial number's length in the remote control (i.e. when no serial number is used) is the option that introduces the largest energy cost in the insulin pump. In this context, though, we take the serial number's length where the total energy cost in the insulin pump is the lowest (i.e. 12-bit), as the patient can not receive the treatment if the insulin pump has no battery.

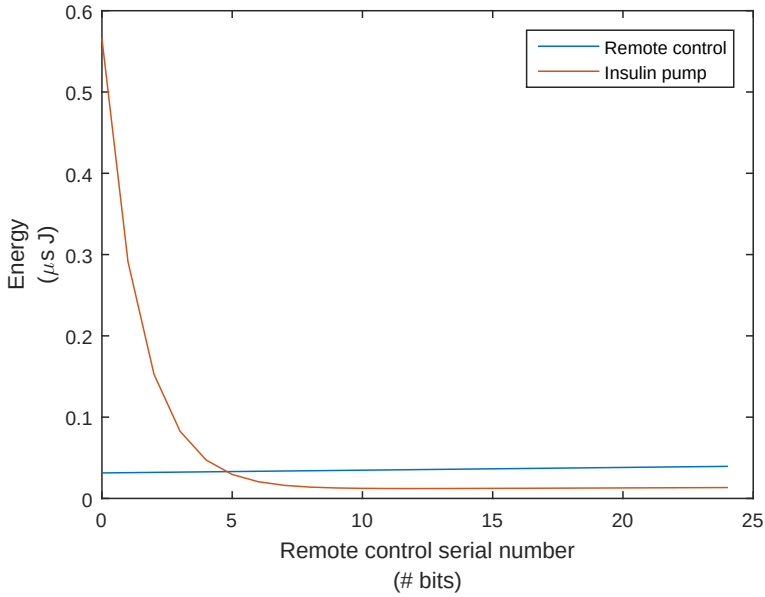

Figure 8: Relation between the energy consumption and the remote control serial number's length for both the remote control and the insulin pump.

As a result, because of using a 12-bit remote control serial number (instead of a 24-bit serial number), the insulin pump does not only receive messages sent by the patient's remote control, but also from other remote controls with the same optimized 12-bit serial number within the communication range. Given the assumptions explained in section 7.2 and assuming a uniformly distributed 12-bit serial number, the expected number of collisions per day is $7800 \cdot (2^{-12})$, namely, 2 collisions.

Optimized solution: Fig. 9 shows the encryption-then-MAC schematic. Initially, the message is encrypted using the encryption key and a fresh counter value. The message contains two fields: the device type and information bits. Subsequently, the 12-bit remote control's serial number (not encrypted), the ciphertext and the counter are used to generate a 64-bit tag, which is appended to the message.

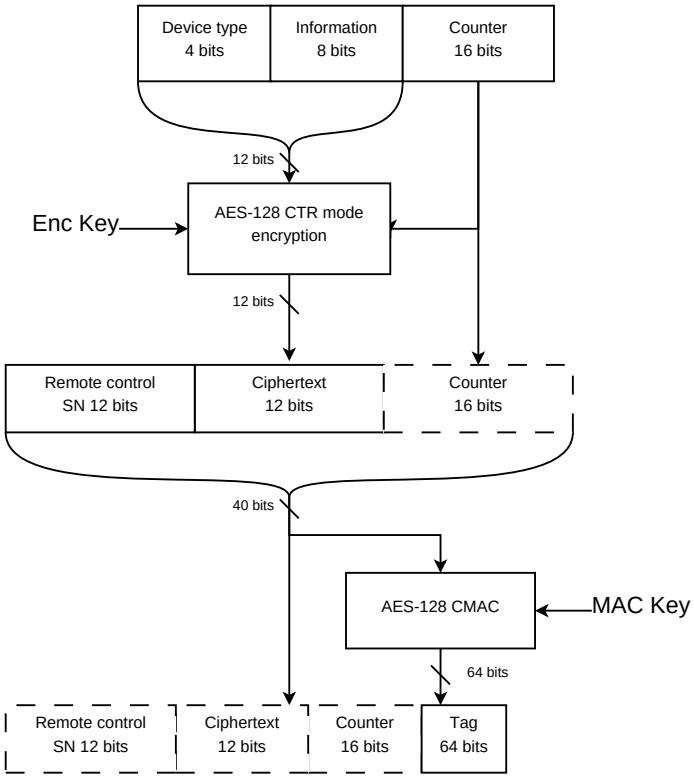

Figure 9: Encryption-then-MAC schematic.

When the insulin pump receives a message from a remote control (not necessarily from the patient's remote control), the former checks (i) whether the 12-bit serial number equals the 12-bit serial number of the valid remote control and (ii) whether the counter is higher than the value of the previously received message. If both conditions are satisfied, the insulin pump verifies the 64-bit tag. If the latter is done successfully, the insulin pump decrypts the message. Otherwise, the message is discarded.

Total energy cost: Because of the message format previously introduced and all the other optimizations, the message length, compared with the original remote control's message format, is increased from 78 to 104 bits. Given the solution that we propose and the assumptions above mentioned, the remote control sends 26 extra bits per message, 150 times per day, which results in an extra communication cost of 8.77mJ. Similarly, the insulin pump receives 26 extra bits per each of the 150 messages sent by the patient's remote control as well as 2 messages sent by other patients' remote control caused by collisions in the serial number. This corresponds to an extra communication cost of 2.92mJ and 0.16mJ, respectively. To encrypt-then-MAC each of these 150 messages, the remote control requires 0.64mJ. Likewise, the insulin pump needs to verify the MAC and then decrypt the 150 messages sent by the patient's remote control and the 2 additional messages sent by other patients' remote controls. To perform these operations, an extra computation cost of 0.65mJ is required.

7.3 Discussion

Table 2 shows the security properties of the different security solutions with their corresponding energy costs for both the remote control and the insulin pump.

Our results show that in all these solutions the computational cost is negligible compared to the cost of transmitting/receiving bits, which evidences the importance of optimizing the message format. On the one hand, we demonstrate that the data confidentiality requirement can be easily accomplished if these devices transmit/receive messages with an optimized format. We show that, by using a reduced message format to lower the communication cost, even adding encryption/decryption still requires a lower energy cost compared to the original design (without security and a non-optimized message format). On the other hand, we show that the authentication requirement is more difficult to accomplish, as this can only be achieved by appending a MAC to the message, which inevitably increases the message length and so the cost of transmitting/receiving bits.

How can the energy costs be further reduced? Our results show that the computation cost is about the 2% of the total energy cost of our final solution. So, even if the cost of computing cryptographic operations could be theoretically reduced to zero, the total energy cost would only be slightly decreased. Therefore, the problem does not solely lie in the design of lightweight cryptographic algorithms but particularly on finding ways to reduce the communication cost, e.g. further optimize the message format previously proposed. Another possibility would be to use a 32-bit tag rather than a 64-bit tag (as currently

used in our solution). However, even though 32-bit MACs are considered to be secure against on-line attacks, attacks where an attacker can capture messages and then try all possible keys off-line could still be carried out. While not practical in this insulin pump system, another option could be to compute the MAC over several messages rather than doing it per every message. Our work clearly shows that there is a need for lightweight integrity solutions which do not significantly increase the message size.

8 Conclusions

In this work we have demonstrated, by reverse-engineering two models of a commonly used insulin pump system, that security through obscurity is a dangerous design approach. Our protocol analysis resulted in the identification of serious security and privacy vulnerabilities. We discovered that no defences against replay and message injection attacks were present, and sensitive patient health-related information was disclosed unencrypted in the wireless communication.

To prevent these active and passive software radio-based attacks, we showed how to secure the wireless link using a security solution based on AES in combination with an optimization of the message format. We evaluated and implemented this solution on a low-cost 16-bit micro-controller similar to the one used in the insulin pump, and compared its security properties and energy costs with alternative solutions that provide less security. We show that there is a need for future research on how to protect the message integrity without largely increasing its length.

In accordance with the principle of responsible disclosure, we have notified the manufacturer six months before disclosure.

9 Acknowledgements

The authors would like to thank Pieter Gillard and Saskia Vanderwegen for their support and the anonymous reviewers for their helpful comments. This work was supported in part by the Research Council KU Leuven: C16/15/058.

Publication

On the (in)security of the Latest Generation Implantable Cardiac Defibrillators and How to Secure Them

Publication Data

Eduard Marin, Dave Singelée, Flavio D. Garcia, Tom Chothia, Rik Willems, and Bart Preneel, "On the (in)Security of the Latest Generation Implantable Cardiac Defibrillators and How to Secure Them," In Proceedings of the 32th Annual Computer Security Applications Conference (ACSAC 2016), 10 pages, 2016.

Contributions

- Principal author. Responsible for the reverse engineering, security analysis and attacks. The proposed defenses are the result of brainstorming sessions with co-authors. Tom Chothia was in charge of the formal analysis of the protocol using ProVerif.

On the (in)security of the Latest Generation Implantable Cardiac Defibrillators and How to Secure Them

Eduard Marin[1], Dave Singelée[1], Flavio D. Garcia[2]
Tom Chothia[2], Rik Willems[3], and Bart Preneel[1]

[1] ESAT-COSIC and iMinds, KU Leuven, Belgium
[2] School of Computer Science, University of Birmingham, UK
[3] Cardiology, University Hospital Gasthuisberg, Leuven, Belgium

Abstract. Implantable Medical Devices (IMDs) typically use proprietary protocols with no or limited security to wirelessly communicate with a device programmer. These protocols enable doctors to carry out critical functions, such as changing the IMD's therapy or collecting telemetry data, without having to perform surgery on the patient. In this paper, we fully reverse-engineer the proprietary communication protocol between a device programmer and the latest generation of a widely used Implantable Cardioverter Defibrillator (ICD) which communicate over a long-range RF channel (from two to five meters). For this we follow a black-box reverse-engineering approach and use inexpensive Commercial Off-The-Shelf (COTS) equipment. We demonstrate that reverse-engineering is feasible by a weak adversary who has limited resources and capabilities without physical access to the devices. Our analysis of the proprietary protocol results in the identification of several protocol and implementation weaknesses. Unlike previous studies, which found no security measures, this article discovers the first known attempt to obfuscate the data that is transmitted over the air. Furthermore, we conduct privacy and Denial-of-Service (DoS) attacks and give evidence of other attacks that can compromise the patient's safety. All these attacks can be performed without needing to be in close proximity to the patient. We validate that our findings apply to (at least) 10 types of ICDs that are currently on the market. Finally, we propose several practical short- and long-term countermeasures to mitigate or prevent existing vulnerabilities.

1 Introduction

Implantable Medical Devices (IMDs) such as pacemakers and Implantable Cardioverter Defibrillators (ICDs) are used to monitor and help control abnormal heart rhythms. ICDs are battery-powered devices that deliver electric shocks to the patient's heart if the heartbeat is too fast. Some ICDs can also act as a pacemaker and give tiny electrical shocks if the heartbeat is too slow. ICDs have evolved over three generations. The first generation (or the oldest) do not have any wireless interface and hence do not allow reprogramming once the ICD is implanted. The second and third generation enable wireless communication with external devices including device programmers and base stations. Device programmers are used by medical personnel to wirelessly modify the ICD's settings or collect telemetry data, whereas base stations, installed in the patients' home, allow remote monitoring by gathering telemetry data from the ICD and sending this data to the hospital. Both device programmers and base stations have a programming head that activates the ICD's wireless interface when it is placed above the implantation site (the patient's chest) for a few seconds.

The second generation of ICDs supports wireless communication between the programming head and the ICD only over a short-range communication channel (less than 10 cm). In the third generation (the latest), the programming head is first used over the short-range communication channel to activate the long range communication link of the ICD. This process is illustrated in Fig 1. Both devices can then communicate with each other over a long-range communication channel (from two to five meters), not requiring the use of the programming head anymore, unless the session expires.

While these advances bring substantial clinical benefits to patients, new security and privacy threats also emerge, specially due to the wireless communication between these devices. Adversaries may eavesdrop the wireless channel to learn sensitive patient information, or even worse, send malicious messages to the ICD. The consequences of these attacks can be fatal for patients as these messages can contain commands to deliver a shock or to disable a therapy.

Our contribution: This paper presents the first reverse engineering and security analysis of the proprietary long-range communication protocol between the device programmer and the latest generation of ICDs. For the reverse engineering we use a black-box approach and inexpensive Commercial Off-The-Shelf (COTS) equipment. This task is not trivial since it was first necessary to find the symbol rate from the waveform of the signals sent by the devices in order to demodulate the captured messages correctly. We show that for proprietary protocols on which we had no prior knowledge or documentation, reverse-engineering is possible by a weak adversary without even needing to have

physical access to the devices. Our second contribution consists of demonstrating several attacks that can compromise the ICD's availability and the patient's privacy. We give evidence that replay and spoofing attacks are possible as well. To evaluate the feasibility of these attacks, we describe several ways to circumvent the short-range communication step, which requires being close to the patient, and perform session hijacking. We validated that our findings apply to (at least) 10 different ICD models. Our third contribution is the proposal of several short- and long-term measures to mitigate or solve the existing vulnerabilities in the latest generation of ICDs including a novel key agreement protocol which we formally verified using ProVerif.

Disclosure of results: In accordance with the principle of responsible disclosure, we have contacted and discussed our findings with the manufacturer before disclosure. Given the sensitive nature of our work, we omitted some of the obtained results to avoid easy replication of the attacks.

Paper outline: The remainder of this paper is organised as follows. Section 1 gives an overview of related work and shows our laboratory setup. Section 2 explains the process of reverse-engineering the proprietary protocol between the device programmer and the ICD. Section 3 describes several strategies to circumvent the short-range communication, which requires close proximity to the patient. Section 4 shows the protocol weaknesses and implementation flaws whereas practical and effective short- and long-term countermeasures to mitigate or solve these vulnerabilities are presented in Section 5. Finally, Section 6 gives concluding remarks.

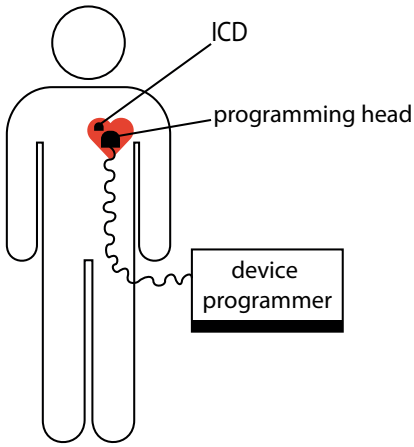

Figure 1: ICD activation procedure.

1.1 Related Work

Software radio-based attacks on IMDs: Several papers have demonstrated that IMDs often lack strong security mechanisms, which makes them vulnerable to different types of remote attacks. Hei et al. showed a simple yet effective attack where adversaries force the IMD to respond to their messages, which reduces the battery life of the IMD [73]. Halperin et al. analysed the proprietary protocol between the device programmer and a second generation ICD to communicate over the short-range communication channel [70]. As no security mechanisms were found, they were able to carry out several software radio-based attacks just by replaying past transmissions sent by the legitimate device programmer. Similar attacks can also be performed on an insulin pump, as shown by Li et al. [86]. Marin et al. fully reverse-engineered the proprietary protocol between all devices in a wireless insulin pump system, and extended the attacks of Li et al. [92]. Unlike the work by Halperin et al. [70], which focused on the short-range communication (less than 10 cm), we analyse the proprietary protocol between the device programmer and a latest generation of ICD over long-range communication (from two to five meters).

Countermeasures: Various countermeasures have been proposed to solve the vulnerabilities found in IMDs. Gollakota et al. presented the "shield", an external device that acts as a proxy between the device programmer and the ICD. The shield jams the messages to/from the IMD to prevent others from decoding them, while still being able to successfully decode them itself [67]. Although this solution mitigates some of the existing problems, it does not protect against adversaries who can transmit malicious messages with much more power than the shield. Tippenhauer et al. demonstrated that the shield does not provide confidentiality as a MIMO eavesdropper could cancel out the interference produced by the shield and then recover the messages sent by the devices [128]. Xu et al. introduced a wearable device, also known as "IMDGuard", which does not only work as a proxy but also performs an authentication process on the ICD's behalf [134]. But Rostami et al. found that the "IMDGuard" is vulnerable to a Man-In-The-Middle (MITM) attack which reduces its effective key length from 129 bits to 86 bits [113]. Rostami et al. presented Heart-to-Heart (H2H), a commitment-scheme-based pairing protocol through which the device programmer authenticates to the IMD without needing to share any prior secrets [114]. H2H implements a novel access-control policy called "touch-to-access" that ensures access to the IMD by any device programmer that can make physical contact with the patient and measure his heart rate. However, Marin et al. found that the H2H is vulnerable to a reflection and a MITM attack [89].

Another line of research relies on exchanging a cryptographic key between the device programmer and the IMD via an auxiliary or Out-Of-Band (OOB) channel. Halperin et al. proposed a zero-power authentication that uses an RFID tag in combination with a piezo-element for audio-based key distribution. However, Halevi et al. demonstrated the feasibility of eavesdropping the audio transmissions of the piezo element [68]. Rasmussen et al. proposed an access control scheme based on ultrasonic distance bounding which enables the IMD to grant access to its resources to only a device programmer that is in its close proximity [109]. However, this typically requires dedicated analog hardware, which makes the solution expensive to integrate in resource-constrained devices like IMDs. Another proposal is to use a Body-Coupled Communication (BCC) channel. Yet, Li et al. showed that remote eavesdropping on a BCC channel is possible with a very sensitive antenna [86]. In this paper, we present practical and effective countermeasures that can be divided into two groups: short-term and long-term measures. The former do not require any modification on the ICDs and hence may be immediately adopted whereas the latter can be implemented in future generations of ICDs.

1.2 Laboratory Setup

Our laboratory setup comprises available Commercial Off-The-Shelf (COTS) hardware including an Universal Serial Radio Peripheral (USRP) [12], a data acquisition system (DAQ) [11] and a few antennas, as shown in Fig 2. In addition, we have the following medical devices: a device programmer, a base station and several ICD models of the latest generation. For our experiments, we created a receiver and a transmitter programs using LabVIEW [7]. The first step of our black-box reverse-engineering approach is to eavesdrop the wireless channel and capture the messages exchanged between the device programmer and the ICD. We then analyse the messages to discover its format, and study how the messages are exchanged between the devices, i.e. the protocol state-machine. Subsequently, we are able to create and send our own messages to the ICD by means of the USRP, the antenna and our transmitter program. To better evaluate the feasibility of these attacks, we also study the ICD activation procedure. For this we use a DAQ and an antenna to intercept the messages exchanged over the short-range communication channel.

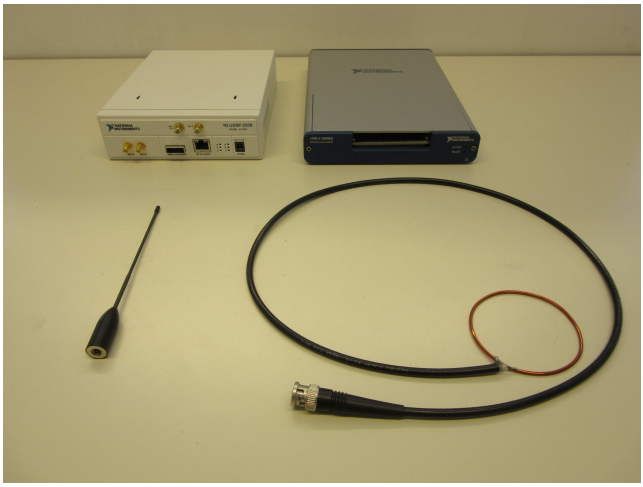

Figure 2: Laboratory setup. At the top, from left to right, are our USRP and the DAQ. Our antennas are shown at the bottom.

2 Intercepting the Wireless Transmissions

Several articles [70, 86, 92] have already pointed out that IMD manufacturers often rely on hiding the protocol specifications to provide security. This is commonly known as security-by-obscurity. Proprietary protocols typically offer very limited or no security guarantees and have been broken via different reverse-engineering techniques. This paper analyses the proprietary protocol between device programmers and the latest generation of ICDs to communicate over a long-range channel. Instead of opening the devices to get their firmware for the purpose of reverse-engineering the protocol, we follow a *black-box approach*. A similar approach has been used in other articles [64,65]. Our black-box approach consists of giving some inputs to the devices and then inferring information by looking at their outputs, i.e. the produced messages. In our work we study the feasibility of reverse-engineering the proprietary protocol by a weak adversary who has limited resources and capabilities. Through meticulous analysis of these messages, we can infer the message format and the protocol state-machine. Our black-box approach, which is a labour-intensive process, is more challenging yet more realistic than other existing techniques, as it assumes a weak attacker who can intercept the messages sent wirelessly using a USRP and an antenna, but cannot have physical access to the devices. We will now summarise our approach and main findings.

2.1 Wireless Communication Parameters

Transmission frequency: The ICD and the device programmer's programming head first communicate over the short-range communication channel (between 30-300 kHz)[4]. After the ICD is activated, both devices communicate over the long-range communication channel using the MICS[5] band (402-405 MHz). The transmission frequency of the devices can be obtained through their Federal Communications Commission (FCC) ID [5].

Modulation: By examining the signals sent by the devices both in the time and frequency domain, we found that the device programmer and the ICD use distinct modulations to transmit their data. In particular, the transmissions from the device programmer to the ICD use a Frequency Shift Keying (FSK) modulation, whereas a Differential Phase Shift Keying (DPSK) modulation is used in the transmissions from the ICD to the device programmer [105].

Symbol rate: Due to the modulations being used, discovering how many symbols (i.e. modulated bits) are sent in each message simply by looking at signal's waveform is a challenging problem.

To estimate the symbol rate, we created a Matlab program that uses the Hilbert transform to obtain the instantaneous frequency of the signal. A key observation is that by demodulating the signals using an FM receiver and looking at the demodulated waveforms, we found that the message sent by the device programmer to request telemetry data is always identical. This message is sent continuously to the ICD when no operation is performed. The first step is to intercept several of these messages and store their waveforms in a file. Our program takes these waveforms as inputs and produces a graph that shows where the frequency shifts occur, i.e. where each symbol starts and ends.

Fig 3 shows the instantaneous frequency of the device programmer's signal. The symbol rate can then be obtained by computing the inverse of the difference between the times where two abrupt peaks occur. However, instead of giving only one symbol rate value, this approach gives a small range of possible values. Therefore, the second step was to create another program that performs a sweep over all possible symbol rate values within this range, increasing the symbol rate by one symbol each time. For each iteration, our program demodulates several of the messages previously captured, and then checks whether the demodulated bits are equal for all the messages. This allows us to find the symbol rate being used by the devices, as the correct symbol rate is the one for which no bit errors are produced.

[4]We do not specify the exact transmission frequency as this may implicitly reveal the manufacturer's identity.

[5]Medical Implant Communications Service.

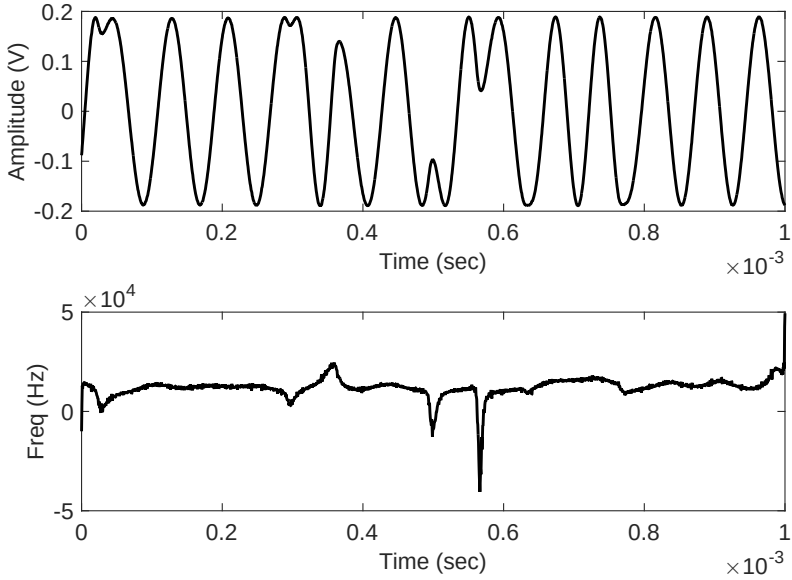

Figure 3: Symbol rate estimation based on the Hilbert transform. In the top chart, the waveform of the signal transmitted by the device programmer. In the bottom chart, the instantaneous frequency of the device programmer's signal.

2.2 Reverse-engineering the Long-range Communication Protocol

In this section, we show how to reverse-engineer the proprietary protocol between the device programmer and the ICD to communicate over the long-range channel. We first activate the ICD and put the device in "interrogation" mode. More details on how adversaries can activate the ICD are given in Section 3.

We found that all messages have a common Start-of-Frame (SoF) that consists of a series of alternating "1s" and "0s" sent consecutively to indicate the presence of an incoming message. This is followed by a preamble sequence which indicates that the information bits are about to begin. To distinguish the messages sent by the device programmer and the ones sent by the ICD, we placed the device programmer close to our USRP while keeping the ICD further away, thus getting more power from the device programmer. Unlike the messages sent by the ICD, which only use one preamble sequence, two preamble sequences can be

used in the messages sent by the device programmer; a specific sequence or its inverse. Messages from device programmers have a fixed length and include a 3-bit End-of-Frame (EoF) sequence whilst ICDs send messages with three possible lengths that do not contain any EoF.

Transmissions device programmer - ICD

We intercepted the messages sent from the device programmer to the ICD while carrying out different operations (e.g. changing the therapy settings). For the sake of simplicity, we will focus only on the messages sent from the device programmer to the ICD in order to change the patient's name. This process typically includes 16 messages and is always composed of two differentiated groups of messages separated by a long message sent by the ICD, as shown in Figure 4. The former group includes messages 1-8, whereas the second group includes the 9th up to the 16th message.

We found that the 16 messages have a x-bit sequence that denotes the message type. In each of the two messages' groups, there are three possible message types independently of the operation being conducted: (i) an opening message, (ii) intermediate messages and (iii) a closing message. We determined that the first and the eighth messages within the second group contain the Serial Number (SN) of the device programmer. The ICD SN appears only in the messages sent by the device programmer when the ICD is in the "no telemetry" mode. In other words, this is sent only if the ICD loses the connection with the device programmer during an ongoing session. Each SN is represented by a 24-bit sequence. Subsequently, we observed that there is a y-bit sequence to indicate the message number within the first group of messages. This field is kept static in the second group of messages. Since the message number field only has a short length and eight messages are sent by the device programmer within the first group of messages, this field is reset frequently. By capturing and analysing the 16 messages sent by the device programmer in several consecutive iterations within a reprogramming session, we found two short counters, in the first and ninth message respectively. Both counters are increased every time an operation is performed and are reset when a new reprogramming session is established.

We discovered that there is a 16-bit sequence at the end of each message that seems to be random and varies depending on the headers and data being sent. This lead us to think that a checksum, such as a Cyclic Redundancy Code (CRC), is used. To validate our hypothesis, we took the GCD of several of these messages (in polynomial form), and discovered that the CRC-16-CCITT is being used [83]. Other mechanisms, such as repetition codes, are used to help the ICD detecting bit errors. We noted that if the patient's name contains less than 14 characters, it is sent three times, otherwise it is sent twice within the first group of messages. Fig 3 shows the device programmer's message format.

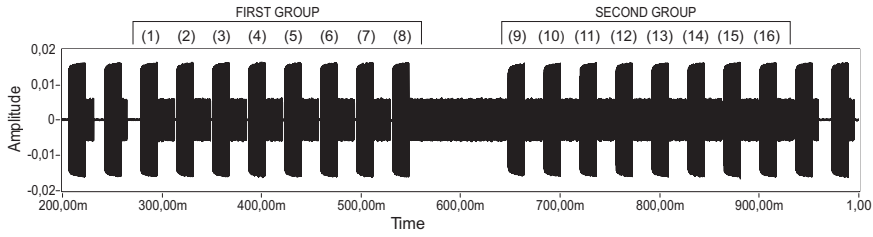

Figure 4: Messages exchanged between the device programmer and the ICD while changing the patient's name.

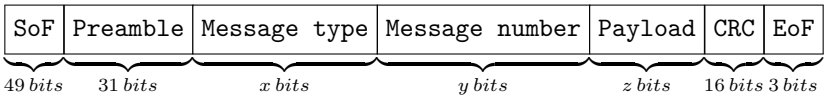

Figure 5: Device programmer's message format. The exact bit lengths are not shown.

Data whitening

We carried out a series of experiments to find how the data is encoded in the message. For this we focused on the messages sent by the device programmer when changing the patient's name.

The first experiment consisted on finding where the letters are within the messages and see how many bits are used to represent each letter. In particular, we changed the patient's name to "A", "AA", "AAA", "AAAA" and "AAAAA", respectively. We then intercepted the messages and compared them with the ones sent by the device programmer when the patient's name field is left empty. We found that the first four letters are sent within the first message and that each letter is represented by an 8-bit sequence. In addition, we observed that there is no unique pattern to represent the "A". The next step was to reprogram the patient's name while keeping a specific letter in more than one position. We modified the patient's name to "AAAA", "ABAB" and "ACAC", respectively. This experiment demonstrated that the way how each letter is encoded depends on its position within the patient's name. In other words, an "A" in the first position is always represented in the same way but differently to an "A" in another position. By comparing the 8-bit sequences of the "A", "B" and "C" in the second and the fourth position, respectively, we noticed that the Hamming distance between the sequences is constant. This allowed

us to conclude that the data is XORed with an output sequence from a Linear Feedback Shift Register (LFSR) (see Figure 6)[6]. The vendor states that this is a data whitening operation to prevent long strings of "1s" and "0s" in the data. However, this operation could also serve as data obfuscation.

In our experiments, we were able to recover the LFSR sequence by intercepting the messages sent by the device programmer when the patient's name is left empty (i.e. only spaces). We then computed the XOR between the first message sent by the device programmer when changing the patient's name to "AAAA" and the LFSR sequence. After performing this operation, we found a unique pattern to represent each of the four "As" of the patient's name. This pattern turned out to be identical to its ASCII representation. Our experiments reveal that this LFSR sequence is constant throughout sessions. Moreover, we found that the LFSR sequence is the same for all the ICDs we studied in our experiments. We validated our findings in 10 different ICD models, and concluded that all models use this technique to encode the data that is sent over the air.

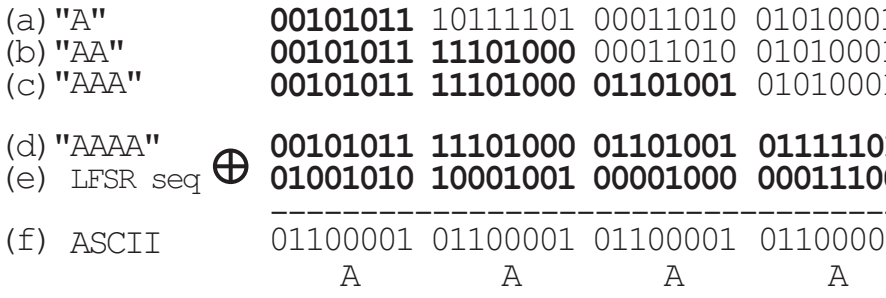

Figure 6: LSFR XOR operation.

Transmissions ICD - device programmer

We intercepted and examined several messages transmitted by the ICD. We did not find any header that is specific for the ICD type or any field that denotes the ICD type. We verified that all messages sent from the ICD to the device programmer use the same LFSR sequence as the one previously described. We noted that all the messages have the ICD SN. In contrast, the ICD includes the device programmer's SN only in replies to no telemetry messages. We discovered that the ICD messages have a counter that helps the device programmer to sort the incoming messages or detect message losses. We observed that most of the

[6]The data and the LFSR sequence that are shown in Figure 6 are not the real ones.

information bits seem random. Since the ICD's leads are no longer connected to the patient's heart and are very sensitive to low-frequency changes, we noticed that they were measuring the ambient noise and treating it as random telemetry data. To investigate where the telemetry data is within the message, instead of injecting our own signal to the ICD's leads, we introduced the ICD's leads into a Faraday cage to isolate them. We then captured several messages sent by the ICD, and noted that they have a more constant pattern which is no longer random. Furthermore, we identified several bit sequences that are common to the three types of ICD message regardless of the operation being performed. These sequences are most likely used for synchronization purposes. Finally, we discovered that, similarly to the messages sent by the device programmer to the ICD, all messages have a 16-bit checksum, which is based on the standard CRC-16-CCITT.

3 How to Activate the ICD?

Before exploiting our findings to carry out attacks, we first need to activate the ICD. To demonstrate the feasibility of these attacks, we describe several ways to bypass the current activation procedure, which requires almost physical contact with the patient and is carried out over a short-range communication channel. For simplicity, in the next sections we often use the term "external device" to denote both device programmers and base stations.

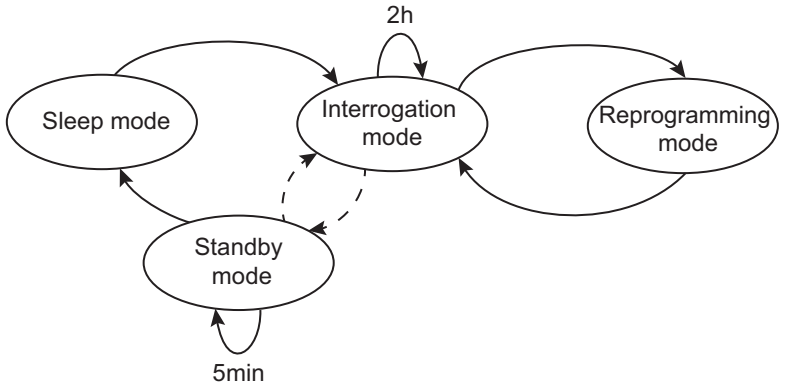

Figure 7: ICD modes of operation.

Our experiments show that the ICD can operate in five different modes: “sleep”, “interrogation”, “reprogramming”, “no-telemetry” or “standby”. In the rest of this paper, we will not discuss the “no-telemetry” mode further since this mode was not relevant for our experiments. Figure 7 gives an overview of the modes. Initially, the ICD is in a “sleep” mode in which it occasionally activates the wireless interface to check whether there is an incoming message sent by a device programmer over the short-range communication channel. Once the ICD is activated, it remains in “interrogation” mode where it continuously sends telemetry data to the device programmer over the long-range communication channel. If no reprogramming operation is performed by the doctor, the ICD is in the “interrogation” mode for two hours. If the doctor modifies the ICD settings within this two-hour window, the ICD switches to “reprogramming” mode for a few seconds and then goes back to the “interrogation” mode, where is kept active for two hours. When the session expires (after two hours), we observed that, instead of immediately switching to “sleep” mode, the ICD goes first to “standby” mode. We will explain the “standby” mode more in detail later in this section.

We will now describe four possible ways to send malicious messages to the ICD, depending on whether the ICD is active, in “standby” or in “sleep” mode.

Exploit an active session: Intuitively, adversaries could attempt to hijack an ongoing session between the external device and the ICD to send malicious commands to the ICD. This is a challenging task since this requires the adversary to be in close proximity to the patient (e.g. in the hospital). Furthermore, adversaries need to send the malicious commands to the ICD while having to block the messages sent by the genuine external device. To masquerade their attacks, adversaries may also send fake telemetry data to the genuine external device to avoid that the doctor/patient notices that the ICD is no longer communicating with it.

Standby mode: We discovered that the ICDs do not immediately switch to “sleep” mode after finishing an ongoing session with the device programmer, but they all remain in a “standby” mode for five minutes. This is a safety feature but also has security consequences.

While being in “standby” mode, any device programmer can activate the ICD again by sending a specific message over the long-range communication channel. This message turns out to be identical for all ICDs. We also found this weakness in the case where the ICD is activated by means of the base station. In that case, the ICD is active for five minutes only if the session with the base station is not terminated correctly.

We were able to impersonate the device programmer and successfully send this message to the ICD to keep it alive. For our experiments, we used the transmitter port of our USRP to emulate the device programmer's behaviour and the receiver port of our USRP to capture this signal and the response sent by the ICD. To distinguish between the messages sent by our USRP and the responses sent by the ICD, we placed the ICD close to the receiver port of our USRP while keeping the transmitter port of our USRP further away, thus getting more power from the ICD. This attack is illustrated in Figure 8. Therefore, adversaries could wait until the session between the device programmer and the ICD finishes and then repeatedly send this message to the ICD. This could be used to drain the ICD's battery, or even worse, to extend the time window as long as needed to send as many malicious messages as required to compromise the patient's safety.

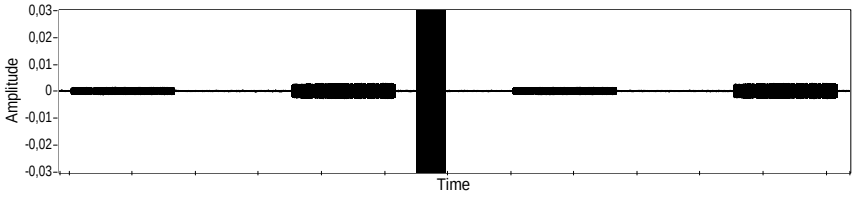

Figure 8: Messages sent to the ICD while the ICD is in "standby" mode in order to activate it. From left to right, two messages sent (with different gain) from our USRP to wake up the ICD, the response of the ICD and two messages sent by the USRP.

Wake up the ICD from "sleep" mode: We noted that the device programmer's programming head is magnetic. To eliminate the possibility that a magnet is needed to bootstrap the communication with the ICD, we conducted an experiment where we placed a magnet near the ICD. The result of this experiment showed that the magnet alone cannot activate the ICD's wireless interface. The next step was to investigate which data is exchanged between the devices before the long-range communication starts. For this we studied the short-range communication between the device programmer and the ICD, focusing on the messages sent by the device programmer.

We used our DAQ and an antenna to capture the messages sent by the device programmer at 30-300 kHz. Every time a new session is established, the device programmer sends three messages to the ICD via the programming head. Following the same steps as those described in the previous section, we were able to unveil the wireless communication parameters being used. In particular, we found that the messages sent by the device programmer are modulated using

a FSK and encoded under Non-Return-to-Zero Inverted (NRZI). In NRZI, a '1' is represented by a transition of the voltage level, whereas a '0' has no voltage transition. We also determined that the symbol rate is 12500 symbols/second.

We created a LabVIEW program to intercept and demodulate the three messages transmitted by the device programmer's programming head. We noted that the first message is always identical regardless of the ICD being used, whilst the second and third message vary depending on the ICD's SN. The other headers and information bits within the second and third message are kept constant, making the short-range communication vulnerable to replay attacks. Thus, adversaries need to eavesdrop the wireless channel only once to intercept the three messages sent by the device programmer. Adversaries could then carry a backpack with all the necessary equipment and re-send these messages to the ICD when the patient is in a crowded place (e.g. the public transportation) where adversaries can be relatively close to the patient and still go unnoticed.

Using legitimate external devices: Alternatively, adversaries can also use any legitimate external device to conduct the attacks. Unlike device programmers, which are big, heavy and cannot be hidden easily, base stations are inexpensive, portable and can be easily purchased. Therefore, one possibility is to use a legitimate base station to carry out these attacks. However, a base station by itself cannot send commands to reprogram the ICD. In our experiments, we show that adversaries can use any legitimate base station to activate the ICD. Since the ICD remains in "standby" mode if the session with the base station is not terminated correctly, adversaries can simply carry the base station in a backpack and turn it off before the communication with the ICD ends in order to keep the ICD alive. Adversaries can then use their own equipment to send malicious messages to the ICD over the long-range communication.

4 Existing Vulnerabilities

In this section we will briefly summarise the weaknesses we found after fully reverse-engineering the proprietary protocol. These weaknesses can result in several types of active and passive software radio-based attacks. We want to stress that adversaries could use sophisticated equipment and directional antennas to extend the distance from which they can carry out attacks by several orders of magnitude.

4.1 Privacy Attacks

Our analysis of the proprietary protocol between the device programmer and one model of the latest generation of ICDs reveals that the messages sent over the air are "obfuscated" using an LFSR sequence. This LFSR sequence is the same for all models that we studied.

The messages exchanged between the devices include patient private sensitive information such as personal data (e.g. his name or medical history) or telemetry data. Clearly, the way they use the LFSR sequence to obfuscate the data can result in serious patients' privacy breaches. Passive adversaries can compromise the patient's privacy just by eavesdropping the wireless channel while there is an ongoing communication. However, this attack typically requires the adversaries to wait until the devices exchange this data. This limitation can be overcome by active adversaries who can additionally send malicious messages to the ICD to request this data.

By intercepting the messages sent by ICDs and looking at their unique SN, adversaries could track, locate or identify patients. For example, adversaries could install beacons in strategic locations (e.g. the train station or the hospital) to infer the patients' movement pattern based on the signals transmitted by their ICDs. This could reveal their addresses, the places they often go, and other potential sensitive information. Furthermore, the messages sent between the devices during a reprogramming session may allow adversaries to infer the patient's treatment or the therapy details. Telemetry data, which is sent continuously by the ICD when it is active, could reveal the patient's health state. Overall, it is clear that the consequences of all these attacks can be severe for patients.

4.2 Denial-of-Service Attacks

As shown in the previous sections, ICDs can operate in four distinct modes: "sleep", "interrogation", "reprogramming" and "standby".

Intuitively, the ICD should immediately switch to "sleep" mode when the communication session with the device programmer finishes or when it expires after two hours with no reprogramming operation. However, we discovered that, after the ICD has been activated, it remains in "standby" mode for five minutes, where it can be put in the "interrogation" mode again if it receives a specific message. This message turns out to be identical for all ICDs and is sent over the long-range communication channel. In other words, there is no need for being in close proximity with the patient to activate his ICD. This is an important

implementation flaw that makes these devices vulnerable to DoS attacks. The purpose of these attacks is to keep the ICD alive by continuously sending this message over the long-range communication, which could drastically reduce the ICD battery life. Yet, this also opens up the door for adversaries to perform other types of attacks more easily, as they can send this message to extend the five minute window as many times as needed to send malicious messages to the ICD without requiring being close to the patient.

4.3 Spoofing and Replay Attacks

After fully reverse-engineering the proprietary communication protocol between the device programmer and the ICD, we were able to fully document the message format in use. Our results show that there is no mechanism to prevent replay attacks; the counters found in the first and ninth message are reset every time a new session is established or after a relatively small number of operations. Without even knowing the protocol specifications, adversaries could successfully perform replay attacks just by re-sending past transmissions sent by the legitimate device programmer. In addition, the protocol does not provide any means to check the integrity and authenticity of the messages. Thus, it is possible to perform spoofing attacks, which allow adversaries to send arbitrary commands to the ICD.

5 Countermeasures

In this section, we present practical and effective countermeasures to mitigate/solve the vulnerabilities found in the previous sections. We divide our countermeasures into two groups: short-term measures and long-term measures. The former group could be deployed immediately to mitigate some of the existing security issues in already-implanted ICDs, whereas the latter group require minor modifications on the devices and hence could be integrated into future generations of ICDs.

5.1 Short-term Measures

Jamming the wireless channel

As shown in the previous section, adversaries can take advantage of the time the ICD is in “standby” mode to carry out a DoS attack, or even worse, to extend the time they can send malicious messages to the ICD. Thus, our

first countermeasure consists of adding a "shutdown" command in all external devices so that they continuously jam the wireless channel while the ICD is in "standby" mode. A more efficient solution is to jam the wireless channel only if an adversary is detected. This is also known as reactive jamming. Several articles have already used friendly-jamming as a defensive mechanism.

One possible drawback of our countermeasure is that it could interrupt the ongoing communications between other legitimate devices. We leverage the fact that the patient typically has his ICD being reprogrammed/interrogated in isolated controlled locations; either in the doctor's office or in the patient's home. This clearly reduces the risks of jamming other ongoing communications. Another downside of our countermeasure is that it works only if the patient stays close to the external device for five minutes while his ICD is in "standby" mode. Due to that the ICD listens to all MICS channels while being in "standby" mode, external devices need to be equipped with several antennas to simultaneously jam all MICS channels.

5.2 Long-term Measures

Adding a shutdown command in the ICDs

Instead of relying on friendly-jamming to prevent adversaries from sending malicious messages to the ICD, our second countermeasure is based on modifying both external devices and ICDs to include a "shutdown" message. This way, the external device can send the "shutdown" message to the ICD before they finish the communication to ensure that the ICD goes directly to "sleep" mode. Even though this countermeasure does not completely solve the existing vulnerabilities, this makes it more difficult for adversaries to send malicious messages to the ICD.

Key agreement protocol

As a long term improvement, Halperin et al. proposed adding standard symmetric key authentication and encryption between the ICD and the programmer. For this they proposed to have the master key on every device programmer (stored in tamper-resistant hardware) and diversified keys in the ICDs. This setup is clearly a significant improvement over existing systems. Yet, having the master key stored in every device programmer is latent risk. If the tamper-resistant hardware of a single device programmer is ever compromised, then there is no way to revoke the keys and every patient with an implant will be exposed indefinitely, or until the IMD is replaced.

Another alternative is to store the master key in the cloud, in order to limit its distribution to a single instance, and have the device programmers online. But this is not a viable option as the device programmers are required to operate (in case of emergency) at all times, including during Internet or cloud provider outages.

In this paper we propose a middle ground between these two approaches: a semi-offline protocol. We leverage the fact that both IMDs and device programmers have a precise internal clock which is synchronised at every communication session. This clock allows the IMD to keep a log file with all critical events and the time when they occurred. Let G_1 and G_2 be two multiplicative groups of prime order q. Furthermore, let $e\colon G_1 \times G_1 \to G_2$ be a bilinear map satisfying:

Bilinearity $\forall g, h \in G_1, \forall a, b \in \mathbb{Z}_q^*, e(g^a, h^b) = e(g, h)^{ab}$.

Non-degeneracy $\forall g \in G_1, g \neq 0$ implies that $e(g, g)$ is a generator of G_2.

Computability e can be computed in polynomial time.

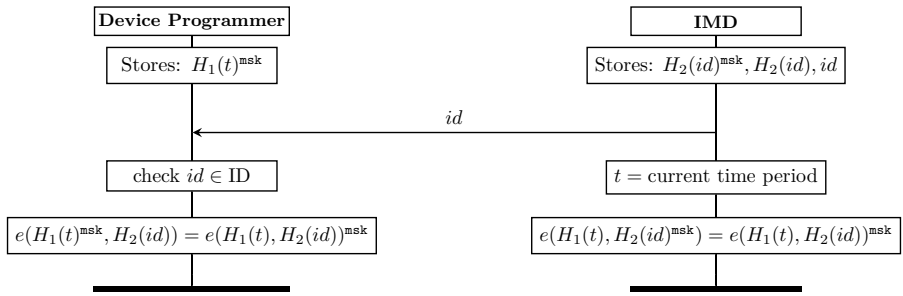

Figure 9: A semi-offline key agreement protocol for IMDs.

Let $H_1, H_2\colon \{0,1\}^* \to G_1$ be two different cryptographic hash functions satisfying standard security requirements.
When the system is initialised, the key generation centre generates the system master secret key `msk` which is stored securely at the back office, and is never shared with anyone. Let ID be the IMD identities domain which is assumed to be disjoint to the time domain. Each IMD $id \in$ ID stores a diversified key $H_2(id)^{\mathtt{msk}}$ which is provided at manufacture time (all the operations here are done modulo q). Device programmers receive a temporal key $H_1(t)^{\mathtt{msk}}$ which is valid to derive all IMD's diversified keys but only for the time period t. This period of time can be anything, but for the sake of example let us take this time period to be three months. In that case, every three months, the

device programmer (or a health-care employee) needs to contact the device manufacturer to obtain the key for the next quarter $H_1(t+1)^{\mathtt{msk}}$ which is sent over a secure channel. In this way, if a device programmer is lost, stolen or tampered with, this can be reported to the device manufacturer and then this device will no longer receive key updates, rendering it useless. Any key material which may have been extracted from the device becomes obsolete after (at most) three months and then the system goes back to a secure state. Fig 9 describes our semi-offline key agreement protocol in detail.

This protocol requires one bilinear pairing computation on the IMD which is expensive, but this only needs to happen once every three months. On a daily basis, IMD and device programmer simply run a standard symmetric key authentication protocol like the one proposed by Halperin et al., using the agreed key $e(H_1(t), H_2(id))^{\mathtt{msk}}$. Note that this protocol does not provide key confirmation, but this can be easily achieved by the symmetric key authentication protocol as it is the case in Halperin et al.

Formal analysis of our protocol

To provide some level of assurance for our protocol we model and analyse it using the applied pi-calculus [35] and the checking tool ProVerif [43]. The applied pi-calculus allows us to model protocols, using primitives such as input, output, new name generation and parallel composition. It also allows us to define functions and equations that can be used to model a range of cryptographic primitives. The ProVerif tool can ensure secrecy and correspondence properties for an arbitrary number of runs of a protocol using a automated theorem proving method, however the tool is not guaranteed to terminate and may report false attacks.

We model an idealised version of bilinear pairings using functions and equations in the applied pi-calculus, i.e., we define the functions $\mathbf{power}(x, y)$, $\mathbf{prod}(x, y)$ and $\mathbf{e}(x, y)$ to represent x^y, xy and the bilinear map $e(x, y)$. We would then like to define the equation:

$$\text{equation } \mathbf{e}(\mathbf{power}(a, x), \mathbf{power}(b, y)) = \mathbf{power}(\mathbf{e}(a, b), \mathbf{prod}(x, y))$$

However, such an equation causes ProVerif's proof tactics to enter any infinite loop. Therefore, we introduce an auxiliary function to represent the right hand side of this equation, i.e., we define $\mathbf{powere}(a, b, \mathbf{prod}(x, y))$ to represent $e(a, b)^{xy}$. We note that this gives an abstract model of bilinear pairings that does not include any number theoretic attacks, such as factoring the product, inverse powers, or low entropy secrets.

Our model is made up of four processes: **Programmer**, which models the programmer protocol role, **IMD** which models the IMD, **CompromisedReader**, which publicly broadcasts a programmers diversified key for a time period different to the one used by **Programmer**, and **CompromisedUnAuthIMD** which models a compromised IMD by publicly broadcasting the diversified key for a medical device that is not one accepted by the programmer.

At the end of their run the **Programmer** and **IMD** processes broadcast a secret value encrypted with the key they have established. We test the system to see if it is possible for an attacker to learn this secret, which would mean they had successfully established a key with the IMD or Programmer. The full model is given in Appendix A.1.

Testing this model in ProVerif we find that it does indeed keep the keys secret. This means that only an IMD with an ID accepted by the programmer, and a programmer with a diversified key for the right time period, can set up and learn keys, even if there are an arbitrary number of old compromised programmers and IMDs.

To check for redundancy in our protocol, and to see what kinds of attacks ProVerif can find, we experiment with possible simplifications. We first try removing the IMD identity check in the programmer (the "**if imdID=id then**" line of the model), in this case ProVerif finds an attack in which the attacker uses a diversified key from an old, compromised IMD. If we also remove the **CompromisedUnAuthIMD** process, we find that the protocol is then safe. This tells us that this identity check by the programmer is only needed to stop attacks using compromised IMDs, if we decided to discount compromised IMDs in our attacker model we would not need this check.

As a second test, we tried using a single hash function, rather than two. In this case, ProVerif finds an attack that lets an attacker impersonate an IMD: The attacker sends the old time stamp from a compromised programmer (t') in place of the id to the targeted programmer. This leads to the key programmer using the key $e(H(t)^{\mathtt{msk}}, H(t'))$, however from the compromised programmer the attacker can learn $H(t')^{\mathtt{msk}}$ and so construct the matching key $e(H(t), H(t')^{\mathtt{msk}})$. This suggests that our protocols use of two hash functions is a sensible precaution to avoid attacks based on confusing times and identities. These additional checks gived us increase confidence that the analysis method we use can find attacks, when present, and that our protocol is not unnecessarily complex.

Differentiating between Device Programmers and Home monitors There is an important distinction to make between Device Programmers and Home monitors as the former are in a much more controlled environment than the latter. Device programmers are not sold to anyone: they are available only to

accredited health-care professionals and institutions whereas home monitors are much more available at the patients home. Some home monitors are sometimes available to purchase on auction sites such as Ebay and is relatively easy to get hold of one. But their usage is also very different. Home monitors only need read access to the IMD in order to forward telemetry information to the relevant health-care practitioner. Therefore, it makes sense to have different keys for each of these devices which provide different access levels. In this way, if the key of a home monitor gets compromised for a period of time, this still represents a potential privacy violation but at least it is not life threatening.

6 Conclusions

In this work we have analysed the security and privacy properties of the latest generation of ICDs. For this we fully reverse-engineered the proprietary protocol between the ICD and the device programmer using commercial and inexpensive equipment. We want to emphasise that reverse-engineering was possible by only using a black-box approach. Our results demonstrated that security-by-obscurity is a dangerous design approach that often conceals negligent designs. Therefore, it is important for the medical industry to migrate from weak proprietary solutions to well-scrutinised security solutions and use them according to the guidelines.

Our work revealed serious protocol and implementation weaknesses on widely used ICDs, which lead to several active and passive software radio-based attacks that we were able to perform in our laboratory. Our first attack consisted on keeping the ICD alive while the ICD is in "standby" mode by repeatedly sending a message over the long-range communication channel. The goal of this attack was to drain the ICD's battery life, or to enlarge this time window to send the necessary malicious messages to compromise the patient's safety. Our second attack aimed at compromising the patient's privacy. For this we leveraged the fact that we were able to recover the LFSR sequence used to "obfuscate" the messages. We discovered that this LFSR sequence is constant throughout sessions and is the same for all ICDs we studied.

We proposed short-term and long-term countermeasures. As a short-term countermeasure, the only solution is to use jamming as a defensive mechanism. As long-term countermeasures, external devices could send a "shutdown" message to the ICD so that the ICD can immediately switch to "sleep" mode after the communication ends. Moreover, we designed and formally verified a semi-offline key agreement protocol between the device programmer and the ICD.

In accordance with the principle of responsible disclosure, we have notified and discussed our findings with the manufacturer before publishing this article.

7 Acknowledgements

The authors would like to thank Stefaan Foulon for his support and the anonymous reviewers for their helpful comments. This work was supported in part by the Research Council KU Leuven: C16/15/058 and the Cryptacus COST Action IC1403.

A Appendices

A.1 A Formal Model of Our Proposed Protocol

```
(* Secure IMD protocol *)
free c.

(* bilinear pairings *)
fun power/2.  (* power(x,y) = x^y *)
fun powere/3. (* powere(a,b,x) = e(a,b)^x *)
fun prod/2.   (* prod(a,b) = a x b *)
fun e/2.      (* e (a^x,b^y) = e(a,b)^(xy)*)

equation e(power(a,x),power(b,y)) = powere(a,b,prod(x,y)).
equation prod(x,y) = prod(y,x).

data one/0.

(* hashes *)
fun H1/1.
fun H2/1.

(* Shared key cryptography *)
fun senc/2.
reduc sdec(y, senc(y,x)) = x.

private free sec.
private free msk.
```

```
(*
Test if the attacker can learn secret
encrypted with the established key
*)
query attacker:sec.
let Programmer = in (c,imdID);
     if imdID=id then
     let rkey = e(rsec,power(H2(imdID),one)) in
     in(c,message);
     out(c,senc(rkey,sec)).

let IMD = let imdkey = e(power(H1(t),one),psec) in
        out(c,id);
     out (c,senc(imdkey,sec)).

let CompromisedReader = new t'; out(c,t');
                        out(c,power(H1(t'),msk)).

let CompromisedUnAuthIMD = new id'; out (c,id');
                           out(c,power(H2(id'),msk)).
process new msk;
        !new t; out(c,t);
        !new id; out(c,id);
    (   let psec = power(H2(id),msk) in !IMD
      | let rsec = power(H1(t),msk) in !Programmer
      | !CompromisedReader | !CompromisedUnAuthIMD )
```

Publication

Securing Wireless Neurostimulators

Publication Data

Eduard Marin, Dave Singelée, Bohan Yang, Vladimir Volskiy, Guy Vandenbosch, Bart Nuttin, and Bart Preneel, "Securing Wireless Neurostimulators," In Proceedings of the 8th Conference on Data and Application Security and Privacy (CODASPY 2018), 12 pages, 2018.

Contributions

- Principal author, except for the evaluation of our randomness source and the design of the antenna.

Securing Wireless Neurostimulators

Eduard Marin[1], Dave Singelée[1], Bohan Yang[1], Vladimir Volski[2], Guy A. E. Vandenbosch[2], Bart Nuttin[3], and Bart Preneel[1]

[1] imec-COSIC, KU Leuven, Belgium
[2] ESAT-TELEMIC, KU Leuven, Belgium
[3] Neurosurgery, UZ Leuven, Belgium

Abstract. Implantable medical devices (IMDs) typically rely on proprietary protocols to wirelessly communicate with external device programmers. In this paper, we fully reverse engineer the proprietary protocol between a device programmer and a widely used commercial neurostimulator from one of the leading IMD manufacturers. For the reverse engineering, we follow a black-box approach and use inexpensive hardware equipment. We document the message format and the protocol state-machine, and show that the transmissions sent over the air are neither encrypted nor authenticated. Furthermore, we conduct several software radio-based attacks that could compromise the safety and privacy of patients, and investigate the feasibility of performing these attacks in real scenarios.

Motivated by our findings, we propose a security architecture that allows for secure data exchange between the device programmer and the neurostimulator. It relies on using a patient's physiological signal for generating a symmetric key in the neurostimulator, and transporting this key from the neurostimulator to the device programmer through a secret out-of-band (OOB) channel. Our solution allows the device programmer and the neurostimulator to agree on a symmetric session key without these devices needing to share any prior secrets; offers an effective and practical balance between security and permissive access in emergencies; requires only minor hardware changes in the devices; adds minimal computation and communication overhead; and provides forward and backward security. Finally, we implement a proof-of-concept of our solution.

1 Introduction

In the US, chronic pain already affects more people than those suffering from diabetes, heart disease and cancer altogether [1]. In recent years, the number of people with movement disorders such as Parkinson's disease or essential tremor has also increased. It is estimated that seven to ten million people worldwide are living with Parkinson's disease [14]. Many of these problems can be relieved with neurostimulators that deliver controlled electrical signals to specific targeted areas in the patient's brain.

The newest generations of neurostimulators often include wireless capabilities that enable remote monitoring and reprogramming through an external device programmer. While the wireless interface enables more flexible and personalized treatments for patients, it also opens the door for adversaries to conduct software radio-based attacks. If strong security mechanisms are not in place, adversaries could send malicious commands to the neurostimulator in order to deliver undesired electrical signals to the patient's brain. For example, adversaries could change the settings of the neurostimulator to increase the voltage of the signals that are continuously delivered to the patient's brain. This could prevent the patient from speaking or moving, cause irreversible damage to his brain, or even worse, be life-threatening.

Beyond attacks that can endanger the patient's safety, there are other attacks that can, for example, compromise the patient's privacy. On the one hand, adversaries could leverage the wireless nature of the communication to intercept the data transmitted over the air. The transmitted data is personal data, and some of it is sensitive medical data. On the other hand, a more sophisticated form of privacy attack would be to use signals extracted from the brain to make inferences about patients. While this is currently not possible, future generations of neurostimulators will use information extracted from patients brain signals for the development of more precise and effective therapies. In that case, adversaries could capture and analyze brain signals such as the P-300 wave, a brain response that begins 300 ms after a stimulus is presented to a subject. The P-300 wave shows the brain's ability to recognize familiar information. Martinovic et al. performed a side-channel attack on a Brain Computer Interface (BCI) while connected to several subjects, and showed that the P-300 wave can leak sensitive personal information such as passwords, PINs, whether a person is known to the subject, or even reveal emotions and thoughts [94]. All the attacks described above clearly show the need for analyzing the security and privacy of neurostimulators.

1.1 Problem Statement and Challenges

Several papers have demonstrated that implantable medical devices (IMDs) with wireless capabilities often lack strong security mechanisms. Halperin et al. were the first to identify some of the potential security and privacy threats on IMDs [69]. Hei et al. proposed simple yet effective denial-of-service (DoS) attacks against IMDs that cannot be prevented with traditional cryptographic approaches. The goal of these attacks is to deplete the IMD's resources such that the battery lifetime is reduced from several years to a few weeks [73]. These attacks are similar to the sleep deprivation torture attack proposed by Stajano and Anderson [127]. In 2008, Halperin et al. analyzed the proprietary protocol between a device programmer and an implantable cardioverter defibrillator (ICD) over a short-range communication channel (less than 10 cm) [70]. Because of the lack of security mechanisms, they were able to realize various attacks by replaying past transmissions sent by legitimate device programmers. Marin et al. fully reverse engineered the proprietary protocol between a device programmer and a latest generation ICD over a long-range communication channel (from 2 to 5 m) [91]. They also showed that it is possible to conduct attacks using only inexpensive hardware equipment without needing to be close to the patient. Similarly, Li et al. were able to emulate legitimate remote controls to perform attacks against insulin pumps [86]. Marin et al. extended the previous work by reverse engineering the proprietary protocol between an insulin pump and all its potential peripherals [92]. In previous work, Denning et al. and Rushanan et al. highlighted the need for evaluating the security and privacy of neurostimulators [56,116]. In this paper, we tackle this problem and investigate the security of the proprietary protocol between a device programmer and a widely used commercial neurostimulator.

Securing the communication between IMDs and device programmers is a non-trivial task. Firstly, IMDs are resource-constrained devices with tight power and computational constraints, and a limited battery capacity. Furthermore, IMDs lack input and output interfaces, such as a keyboard or a screen, and cannot be physically accessed once they are implanted. Secondly, IMDs need to satisfy several important requirements for proper functioning, such as reliability, availability and safety. Adding security mechanisms into IMDs is challenging due to inherent tensions between some of these requirements and the desirable security and privacy goals. For example, IMDs should provide permissive access control policies such that doctors can access the IMD to reprogram it or extract patient's data from it in emergencies. A small delay when contacting care providers for device-specific cryptographic keys could prevent the patient from receiving care on time. However, if access control policies are not sufficiently strong, IMDs could be exposed to attacks. This example clearly shows the need for designing security solutions that achieve a trade-off between security and

permissive access control in emergencies.

To bootstrap a secure communication channel between the IMD and the device programmer, cryptographic keys first need to be securely initialized and shared. This can be achieved using traditional approaches based on symmetric or public-key cryptography. For example, one could pre-install a device-specific symmetric key in each IMD during manufacturing, or use a key diversification protocol [70,91]. However, as explained above, it may not be possible for medical personnel to contact care providers on the fly for requesting device-specific cryptographic keys. If a key diversification protocol is used, storing the master key in tamper-proof hardware in all device programmers would bring substantial security risks; if the master key is ever compromised, the security of millions of IMDs would be at risk. Instead, one could opt for storing the master key in the cloud but this is not a viable option since device programmers are required to operate at all times, including during Internet or cloud provider outages. Another solution would be to pre-install a public-private key pair in each IMD, and use any secure key transport or key agreement protocol. This approach would require a robust, worldwide infrastructure that keeps an updated list of trustworthy device programmers as well as a revocation mechanism that ensures that IMDs cannot communicate with untrustworthy device programmers [36]. Unfortunately, having such a robust worldwide infrastructure is difficult, and IMDs do not have an Internet connection to periodically get a certificate revocation list (CRL) or sufficient memory to store all the necessary certificates. This shows that traditional solutions may not be applicable for IMDs due to their unique functional requirements and limited resources.

1.2 Contributions

To the best of our knowledge, this is the first paper that evaluates the security of the wireless communication protocol between a device programmer and a widely used neurostimulator from one of the leading IMD manufacturers. In addition, we propose a practical and efficient security architecture that enables the device programmer and the neurostimulator to create a secure communication channel. In detail, our main contributions can be summarized as follows:

- **Security analysis**. We describe the process of how to reverse engineer the proprietary protocol between the device programmer and the neurostimulator. We follow a *black-box reverse engineering* approach and use inexpensive commercial hardware equipment.
- **Software radio-based attacks**. We assess the feasibility and demonstrate software radio-based attacks on neurostimulators. Furthermore, we

elaborate on how adversaries can overcome some of the limitations and challenges of these attacks.

- **Low-cost source of randomness for neurostimulators**. We explore the possibility of using a patient's brain physiological signal as a randomness source for generating keys, and detail the process of extracting entropy from it. We evaluate our technique using real data extracted from the brain of 22 mice.
- **Security architecture for neurostimulators.** We present a complete security architecture that includes the generation of random session keys, the secure transportation of these keys from the neurostimulator to the device programmer and the cryptographic protocols to secure the communication flow.

Disclosure of results. We followed the principle of responsible disclosure and contacted the manufacturer six months before publishing our results. After discussing our findings with the manufacturer, we chose not to disclose the full details of the obtained results since this could help someone to mount these attacks on neurostimulators used by patients.

Paper outline. The remainder of this paper is organized as follows. Section 2 gives an overview of related work. Section 3 describes the devices that comprise the neurostimulation system, whereas our laboratory setup is shown in Sect. 4. Section 5 details the process of reverse engineering the proprietary protocol between the device programmer and the neurostimulator to discover the message format and the protocol state-machine. Section 6 shows several software radio-based attacks that we are able to conduct on the neurostimulator, while a security architecture to preclude these attacks is proposed in Sect. 7. Section 8 discusses possible limitations of our solution and provides some directions for future work. Section 7 gives concluding remarks.

2 Related Work

There are two main categories of countermeasures to secure the communication between the IMD and the device programmer: (i) those based on using an external device as a proxy, and (ii) those relying on an out-of-band channel to allow the devices to securely agree on a cryptographic key.

2.1 External Devices

Gollakota et al. proposed an external device known as "shield", that jams the messages to/from the IMD to prevent others from decoding them, while still being able to successfully decode the messages itself [67]. While the shield can mitigate some of the existing security problems, it offers only very limited protection since adversaries could bypass it by transmitting messages with more power than those sent by the shield. Furthermore, a multiple-input multiple-output (MIMO) eavesdropper could suppress the jamming signal and recover the data sent by the devices, as shown by Tippenhauer et al. [128]. Xu et al. presented the "IMDGuard", a wearable proxy device that performs an authentication process on the ICD's behalf [134]. The IMDGuard relies on patient's electrocardiography (ECG) signals to generate a symmetric key that is valid for one session and is known only to the IMD and itself. However, Rostami et al. found that the IMDGuard is vulnerable to a man-in-the-middle (MiTM) attack which reduces its effective key length from 129 bits to 86 bits [113]. Overall, although external devices that use friendly jamming could help protecting legacy devices, they could also jam transmissions sent by legitimate devices. In some countries, jammers are illegal and their use can result in large fines. Thus, it is unlikely that these solutions will be accepted by the US federal drug administration (FDA) and adopted by manufacturers.

2.2 Out-of-band Channels

Another prominent family of countermeasures relies on establishing a cryptographic key between the device programmer and the IMD via an auxiliary or out-of-band (OOB) channel. OOB channels typically have low-bandwidth and are easy to set up. There exist two types of OOB channels: (i) *authentic* or (ii) *secret*. The former represents a channel where Bob is guaranteed that the message he receives was actually sent by Alice. The data can however be eavesdropped by others. The latter represents a channel where Alice is guaranteed that the message she sends is only received by Bob. However, Bob does not know that the data comes from Alice.

Halperin et al. proposed to use a radio-frequency identification (RFID) tag in combination with a piezo-element to achieve audio-based key distribution [70]. They were the first to introduce a countermeasure that does not consume energy from the battery. Nevertheless, the audio transmissions generated by the piezo-element can be eavesdropped, as shown by Halevi et al. [68]. As this technique uses a carrier frequency within the audible range so that it can be perceived by patients, it is unclear whether this technique could be applied in noisy environments. Kim et al. presented a vibration-based key transport

protocol through which the device programmer and the IMD can agree on a cryptographic key [80]. Since vibration results in unintentional acoustic emanations, the authors proposed that the device programmer generates a masking sound to hide the acoustic emanations. However, it requires the IMD to have extra hardware, and it is unclear whether the masking sound produced by the device programmer can have an effect in the data sent over the vibration channel. Furthermore, Trippel et al. showed that the integrity of audio and vibration sensor outputs can be compromised by injecting malicious analog acoustic signals [129]. This could allow adversaries to modify the key being sent over the audio/vibration channel in their favor so that the IMD establishes a key with a malicious device. Rasmussen et al. proposed an access control scheme based on ultrasonic distance bounding which enables the IMD to accept any device programmer that is in its close proximity [109]. The main limitation of this solution is that it requires dedicated analog hardware, which makes it not suitable for IMDs. Unfortunately, all the previous solutions have important limitations and drawbacks or have been proven to be vulnerable to security attacks.

The closest solution to ours is the heart-to-heart (H2H) protocol proposed by Rostami et al. [114]. H2H uses a novel access control policy called "touch-to-access" which ensures access to the IMD by any device programmer that can touch the patient's skin to measure his interPulse Interval (IPI) (i.e. the time between heart beats). In H2H, both the device programmer and the IMD need to simultaneously measure the patient's heart rate. The heart rate is then used as a "fuzzy password" that is known only to the device programmer and the IMD. The authors state that one of the main advantages of using the patient's heart rate is that it can be measured anywhere in the patient's body just by touching his skin. However, this solution has several weaknesses and limitations. Firstly, Marin et al. showed that H2H is vulnerable to a reflection and a MiTM attack [89]. Secondly, it is unclear how the authors transformed the IPI signal from analog to its digital form. Thirdly, the H2H protocol has a large communication and computation cost. It requires the IMD to transmit a substantial amount of messages and uses public-key cryptography, which can be too energy consuming for resource-constrained devices such as IMDs. Lastly, several articles have shown that the IPI can be measured remotely using a wide range of techniques. For example, Calleja et al. showed that it is possible to remotely gather information of cardiac signals using widely available inexpensive hardware [47]. Seepers et al. evaluated the feasibility of remote attacks using the existing remote photoplethysmography (rPPG) methods [122]. Their evaluation demonstrates that rPPG achieves similar accuracy as when measuring the heart rate by touching the patient's skin. Poh et al. proposed a simple, low-cost technique through which it is possible to measure the heart rate using a standard webcam [103]. Recently, Katabi et al. demonstrated that WiFi

signals can be used to detect the breathing and heart rate of individuals [24]. All these papers render H2H (and other systems relying on the secrecy of the patient's heart rate such as the IMDGuard) insecure. In this paper, we propose a practical and effective solution that follows the "touch-to-access" principle, but overcomes all the previous limitations.

3 Neurostimulation System

This section describes the devices that comprise the neurostimulation system. This includes device programmers used by doctors and patients as well as neurostimulators.

Device programmers: they are external portable devices used to read out patient's medical data stored on the neurostimulator or to reprogram its settings. There are two types of device programmer: those used by doctors and those used by patients. The former has full privileges to read data and modify the neurostimulator's settings, whereas the latter has restricted permissions, allowing only controlled therapy modifications, as established by the doctor.

Neurostimulators: they are devices implanted subcutaneously near the clavicle that are connected to the brain through several leads. They have limited memory storage, processing power and a battery with limited capacity that cannot be recharged or replaced. When the battery is depleted, the patient needs to undergo surgery in order to get a new implant. Such a surgery always introduces a small, but not negligible risk of infections, sometimes even with lethal consequences. We deliberately chose not to disclose how long the battery lasts because this could implicitly reveal the manufacturer and neurostimulator models that we investigated.

The device programmer contains a magnetic programming head that is used to communicate with the neurostimulator over a wireless bi-directional short-range communication channel (less than 10 cm). The programming head needs to be placed on the patient's chest in close proximity to the neurostimulator for the entire duration of the communication. In most cases, patients have their neurostimulators being interrogated and/or reprogrammed in isolated controlled locations, e.g. in the doctor's office. However, patients also need to frequently use their own device programmer in order to adjust the stimulation configuration, for example, when lying, sitting or walking.

4 Laboratory Setup

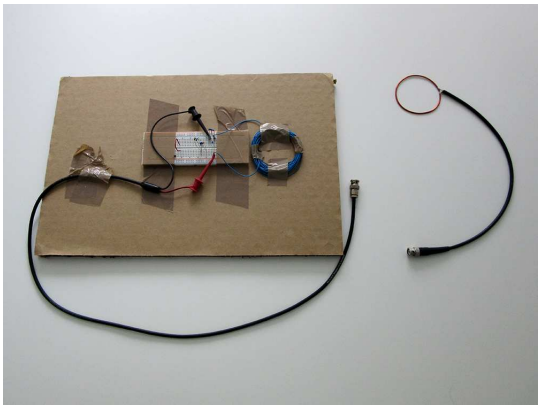

Figure 1: Antennas used for receiving and transmitting signals. The transmit antenna is shown on the left, the receive antenna on the right.

Our laboratory setup comprises inexpensive commercial hardware devices including a standard laptop and a USB-6351 data acquisition system (DAQ) from National Instruments [11]. Moreover, we built two antennas for receiving and transmitting messages respectively, as shown in Fig. 1. For receiving messages, we created a simple antenna by cutting a coaxial cable and connecting a circular piece of copper to it. Even though this antenna allowed us to capture messages exchanged between the devices, its impedance was too low to transmit signals with enough power. To overcome this problem, we designed our own antenna for transmitting signals (for more details, we refer the reader to Appendix A.1).

5 Intercepting RF Transmissions

This paper analyzes the security of the proprietary protocol between the device programmer and the neurostimulator to communicate wirelessly over a short-range communication channel. This is a challenging task because there is no information available about the protocol specifications. IMD manufacturers typically rely on keeping the protocol specifications secret as a means to provide security (i.e. security-through-obscurity). Protocol reverse engineering implies finding both the message format and the protocol state-machine without knowing the protocol specifications. Several articles [70, 91] have already shown that proprietary protocols can be reverse-engineered. While several techniques can

be applied to reverse engineer proprietary protocols, physical access to the devices is often necessary, e.g. to extract the firmware.

In this paper, we follow a *black-box reverse engineering* approach which allows us to find the inner-workings of the protocol by only providing inputs to the devices and looking at their outputs. In other words, we change any of the neurostimulator's settings using the device programmer, and then inspect the format of the transmitted messages. Our black-box methodology is a labor-intensive and challenging process yet it is more realistic than other approaches. By following this approach, we assess the feasibility of reverse engineering the proprietary protocol by a weak adversary with limited resources and capabilities who cannot have physical access to the devices but can only intercept the messages sent wirelessly. We acknowledge that having access to the device programmer during our experiments in order to perform certain actions can speed up the process of reverse engineering the protocol. However, although this process may take longer when the actions being performed on the device programmer are not known, adversaries can still learn the inner-working of the protocol by intercepting and analyzing transmissions sent over the air by legitimate devices.

Next, we describe how to reverse engineer the proprietary protocol used by a specific neurostimulator model. However, we also conducted various experiments using other neurostimulator models, and came to similar findings as the ones described in this paper. Figure 3 shows the format of the messages sent by the device programmer.

5.1 Wireless Communication Parameters

Before capturing the messages exchanged between the device programmer and the neurostimulator, we first needed to discover several wireless communication parameters such as the transmission frequency, modulation and encoding scheme, and the symbol rate being used. This step is crucial as the use of slightly different parameters compared to those used by the devices would result in an incorrect demodulation of the captured messages, i.e. messages with erroneous bits.

The first step of our analysis was to find the frequency at which the devices transmit their messages. By entering the unique device programmer's federal communications comission identifier (FCC ID) in the FCC database [5], we determined that these devices transmit their messages at 175 KHz. We then captured several messages sent by the devices, and visualized the signals in the time domain. The waveform of these signals indicates that an on-off keying (OOK) modulation scheme is being used, as illustrated in Fig. 2. In an OOK modulation, the presence of a carrier wave is used to indicate a binary '1' and its

absence indicates a binary '0'. Similarly to passive RFID devices, all messages are encoded to ensure that the neurostimulator receives enough power from the device programmer even when a long string of "0s" is sent. Encoding of '1' or '0' is based on a variable width high-pulse and a fixed width space. For security reasons, we decided not to reveal the symbol rate being used.

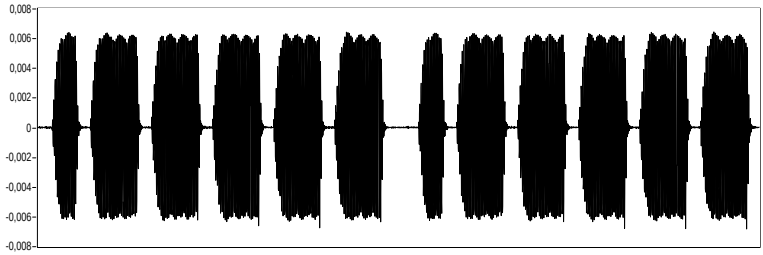

Figure 2: Waveform of a signal transmitted by the device programmer.

5.2 Reverse Engineering the Proprietary Protocol

We intercepted messages exchanged between the two types of device programmer and several models of neurostimulator. However, we focused on analyzing the messages sent by the device programmer since understanding these transmissions would allow us to emulate a legitimate device programmer. (In Sect. 5.3 we show that most messages sent by neurostimulators are simply acknowledgements). After demodulating the captured messages, we observed that they all include a common synchronization sequence (not shown in Fig. 3) that consists of a series of alternating '1s' and '0s'. Subsequently, we found several message fields including headers, information data and checksums.

As a first experiment, we grouped the messages sent by the doctor's device programmers and patient's device programmers, respectively, in two different clusters. This experiment allowed us to determine that there is a 16-bit sequence at the beginning of each message which can take two values depending on the type of device programmer being used. Similarly, we compared the messages sent by each different device programmer, and discovered that there is a unique 16-bit sequence that varies depending on the device. This led us to conclude that this field represents the unique serial number (SN) of each device programmer. Following this approach, we also found two 16-bit sequences that denote the model and SN of the neurostimulator, respectively.

At the end of some messages, there are two 16-bit sequences that seem to be uniformly distributed. This made us think that they could be checksums to

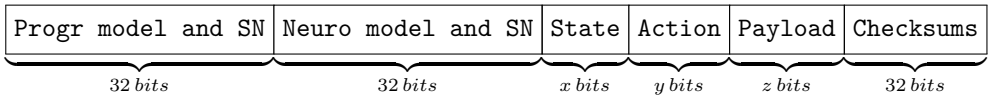

Figure 3: Device programmer's message format.

detect or correct bit errors. Our first hypothesis was that these sequences correspond to a cyclic redundancy check (CRC). This is because CRCs are widely used, easy to implement and very effective at detecting bit errors. Since CRCs are linear functions, we first checked whether the linearity property holds for the second 16-bit checksum. For this purpose, we computed the XOR of two messages and verified whether the result produces a valid message. As the linearity property was satisfied, the second step was to find all the CRC parameters such as the polynomial, initial XOR value, final XOR value as well as whether the input and/or the output is reflected. To discover these parameters, we created a program that brute-forces all possible combinations of CRC parameters. This task required less than 5 minutes on a standard laptop. Following the steps described above, we discovered that the second 16-bit checksum corresponds to a CRC checksum that is computed over the entire message using the standard CRC-16-CCITT [83].

We then repeated this approach for the first 16-bit checksum but we did not succeed on finding the CRC parameters. A key observation was that this checksum remains identical when performing a specific action regardless of the device programmer and/or neurostimulators being used. In other words, this checksum depends only on the state, action and payload fields. To create messages with a valid checksum, we intercepted several messages sent by the device programmer, and XORed them to create new messages where only one bit is set to '1' and the rest of bits are set to '0'. We repeated this approach for each of the bits within these message fields, and created a code-book with all possible messages and their corresponding checksums. Let us give an example to describe how to compute the first 16-bit checksum of a message using our approach. Assuming that our message contains a '1' in the third, fifth and eleventh position, then we can compute this checksum by XORing the checksums of the third, fifth and eleventh messages, respectively, within our code-book.

5.3 Protocol State-machine

This section describes the three phases of the communication between the device programmer and the neurostimulator. This includes the initialization phase, the reprogramming phase and the termination phase.

Initialization phase: Initially, both devices exchange several messages so that the device programmer requests all the information stored on the neurostimulator.

We found that the device programmer always starts the communication by sending a message that is identical across sessions which contains the model and serial number of the device programmer. Within this message, the fields that denote the SN and model of the neurostimulator are kept empty. This is because the device programmer is not linked to any neurostimulator and does not yet know with which neurostimulator it is communicating. The neurostimulator then replies with a message containing its SN and model. From that point until the end of this phase, the device programmer always sends one message to request data whereas the neurostimulator replies back with two distinct messages. The former is an acknowledgment that indicates whether the message was received correctly, while the latter contains the data.

Reprogramming phase: After the initialization phase, the device programmer can be used to modify the neurostimulator's settings as many times as needed within the same protocol session. We discovered that the device programmer sends only one message to adjust the settings, to which the neurostimulator replies with two different messages that remain identical regardless of the action being performed.

Termination phase: Before an ongoing communication is terminated, the device programmer sends a message to the neurostimulator which enables the latter to switch to power saving mode. The neurostimulator replies back with the same two messages that it sends after it has been reprogrammed.

In the next section, we show that it is not necessary to follow the normal protocol flow in order to perform attacks on the neurostimulators.

6 Software Radio-based Attacks

This section details several software radio-based attacks that we are able to perform on the neurostimulator which could endanger the safety and compromise the privacy of patients. We conducted all these attacks in an isolated laboratory

environment with the device programmer turned off. We focus on an experiment where we changed the patient's name (programmed on the neurostimulator) since this allows us to easily verify whether the experiment succeeded. However, we want to emphasize that we can mimic the behavior of a legitimate device programmer to send valid messages to a neurostimulator in order to change any of its settings.

Replay attacks: We were able to modify any of the neurostimulator settings just by intercepting and replaying past transmissions sent from legitimate device programmers. However, this attack has two important practical limitations. The adversary needs to wait until there is an ongoing communication between a device programmer and a neurostimulator in order to intercept the messages. Furthermore, the adversary can only replay messages that were already transmitted by legitimate device programmers, which clearly limits the impact of the attack.

Spoofing attacks: Unlike the previous attack, spoofing attacks require to have partial or full knowledge of the protocol. After reverse engineering the protocol, we were able to create any arbitrary message and send it to the neurostimulator. One possible limitation that makes this attack less practical is that the adversary needs to know the neurostimulator's SN in order to create a valid message. Intuitively, adversaries could obtain the neurostimulator's SN by intercepting the exchanged messages while there is an ongoing communication. As the first message sent by the device programmer is always identical regardless of the neurostimulator, adversaries could also replay this message to a neurostimulator and intercept the response since this contains its SN.

We found a way to overcome even this limitation, which allows adversaries to send messages to a neurostimulator without knowing its SN. More specifically, we discovered that neurostimulators accept messages with an empty neurostimulator SN field. In our experiments, we were able to change the patient's name, programmed in the neurostimulator, by sending a single message with no neurostimulator's SN. The impact of this attack is quite large as an adversary could create a valid message, without a SN, and reuse it for all neurostimulators. The only challenge for adversaries is to be close enough to the victim in order to communicate to the neurostimulator. However, there are definitely several scenarios, such as a crowded subway, where this would be possible.

Privacy attacks: Since our reverse engineering on the proprietary protocol shows that the data sent over the air is unencrypted, passive adversaries can eavesdrop the wireless channel to infer private information about patients. Active adversaries can additionally send messages to the neurostimulator to request specific data. The data sent over the air between the device programmer and the neurostimulator includes diagnosis, symptoms, disease or therapy

information. Moreover, all messages exchanged between the devices include a unique neurostimulator SN which could be used for adversaries to identify, locate and track patients. For this purpose, adversaries could install beacons in strategic locations (e.g. in hospitals or train stations) to learn the patients' movement pattern based on the signals transmitted by their neurostimulators.

DoS attacks: One would expect that neurostimulators switch to "sleep mode" once an ongoing communication is terminated. However, we found that neurostimulators always remain active and accept a message as long as its format and checksums are correct. We also observed that adversaries can send valid messages to the neurostimulator without needing to first execute the initialization phase. While we did not try to quantify the effect of performing a DoS attack against these devices, adversaries could repeatedly send malicious messages to the neurostimulator in order to deplete its resources faster, similarly to the attacks proposed by Hei et al. [73].

7 Security Architecture

In this paper, we propose a practical security architecture that allows the device programmer and the neurostimulator to securely exchange messages. Our solution consists of three main items: (i) session key initialization, (ii) key transport and (iii) secure data communication. The first two items are particularly challenging for medical implants.

To bootstrap a secure communication channel, a symmetric (session) key need to be shared between the devices. This key could be generated by the device programmer and transported from the device programmer to the neurostimulator using an OOB channel. However, this approach presents two limitations. Most OOB channels, that allow to transport a key from the device programmer to the IMD, are shown to be vulnerable to security attacks or require to add extra hardware components in the neurostimulator. An alternative solution would be to generate the key in the neurostimulator and transport it from the neurostimulator to the device programmer. Halperin et al. presented a audio-based key transport solution where the IMD generates a random symmetric key and transports it to the device programmer through an acoustic channel [70]. Even though the authors stated that the key can be received only by a device programmer that can make physical contact with the patient, Halevi et al. showed that it is possible to recover the key from far away by eavesdropping the acoustic channel [68].

We propose a solution where a symmetric (session) key is also generated in the neurostimulator and transported from the neurostimulator to the device

programmer using a secret OOB. Our solution overcomes the previous limitations and ensures that the key can only be picked up by a device programmer that touches the patient's skin long enough. Below, we define our adversarial model and give an overview of all the building blocks of our solution.

7.1 Adversarial Model

We consider the presence of strong adversaries who can eavesdrop or jam the wireless channel, as well as modify, replay or forge messages. Adversaries can be in close proximity with the patient and can possess any legitimate device programmer, even those that already interacted with the neurostimulator. However, adversaries cannot compromise the neurostimulator or the device programmer while being used, since this would make it impossible to protect the neurostimulator. Doctors are trusted and do not collude with adversaries. Adversaries can neither touch the patient's skin long enough without the patient noticing it, nor compromise any device that can make physical contact to the patient (e.g. a smart watch). Within the community, it is accepted to assume that physical contact to a patient means the ability to cure or harm [114]. When designing security solutions and defining the adversarial model for medical devices, it is important to note that the primary goal of a medical device is to treat patients. Clearly, safety is of utmost importance and security should never interfere with this task.

7.2 Overview of the Solution

Our solution involves three main steps: (i) key generation, (ii) key transport and (iii) secure data exchange. More concretely, our solution works as follows. Firstly, the neurostimulator uses a physiological signal from the patient's brain as a source of randomness for generating a 128-bit symmetric key. This key is valid only for a single session and is independent from old and future keys. Thus, if an adversary ever compromises a session key, he will not be able to compute past and future keys, i.e. backward and forward security are guaranteed. Secondly, this session key is transported securely and reliably from the neurostimulator to the device programmer through a secret OOB channel. Similarly to H2H [114], our solution follows a "touch-to-access" access control policy where access to the neurostimulator is granted only to device programmers that can touch the patient's skin long enough. This provides a practical yet effective balance between security and permissive access in emergencies since it allows medical personnel to have immediate access to the neurostimulator without needing to contact care providers for device-specific cryptographic keys. In contrast

to the existing solutions, key transportation can be achieved without adding extra hardware components on the neurostimulator. Once the key has been transported, the devices can optionally run a standard authentication protocol so that they can prove to each other that they posses the session key. Finally, this session key is used to bootstrap a secure communication channel between the devices.

7.3 Key Generation

True random number generators (TRNGs) are an essential building block of cryptographic systems for generating cryptographic keys, nonces and masks. Unfortunately, IMDs typically use microcontroller-based platforms and lack dedicated TRNGs, making it non-trivial to generate random numbers on these devices. Several articles have proposed to use the initial content of the static random-access memory (SRAM), on-chip RC oscillators and on-board external clocks as an alternative to TRNGs [74, 76]. However, these approaches need to be evaluated for each specific device since their performance depends on the platform and hardware components being used.

Recently, the use of physiological signals extracted from the patient's body has been proposed for generating cryptographic keys. This approach has several advantages with respect to traditional approaches based on PRNGs or TRNGs. Specifically, they allow to reuse signals that are already being gathered by the devices as a low-cost source of randomness. A possible limitation is that some physiological signals, such as those extracted from the patient's heart, can be predicted or measured remotely, which makes them vulnerable to attacks. In this paper, we explore the potential of using a signal from the patient's brain, that cannot be measured remotely by adversaries, as a randomness source for generating cryptographic keys. Our approach can be applied to any medical device that can measure the LFP signal from the patient's brain.

LFP signal as a randomness source

We propose to use a physiological signal from the patient's brain called *local field potential* (LFP), which refers to the electric potential in the extracellular space around neurons. The choice of the LFP for randomness extraction is based on the following reasons. Firstly, neurostimulators can easily measure signals in the patient's brain. There are already neurostimulators on the market that use the LFP to create feedback for the delivered stimulation. As the LFP can be collected with the existing lead configurations (i.e. without changing the leads' position or requiring extra leads), future generations of neurostimulators will require only minor software changes to be able to measure this signal. Secondly, unlike other physiological signals such as the ECG, the LFP can be measured

only through direct contact with the patient's brain, thus cannot be obtained remotely.

To analyze the feasibility of extracting randomness from the LFP to generate a 128-bit key in the neurostimulator, we used real LFP data collected from 22 mice[4]. Figure 4 shows LFP data collected from one of the 22 mice. While we recorded LFP data simultaneously from 16 electrode contacts connected to the mice brain using the W16 wireless recording device [17], we only looked at one of these channels. Our technique to extract randomness can be applied to LFP data collected from any of the 16 different channels. During each LFP recording, the mice were walking on a horizontal ladder. This is a well-known test that is used for many different purposes (for more details about this test, see Metz and Wishaw [95]).

The first step in our assessment was to extract the least significant bit of the sampled LFP data and used it as the raw random number. This is because the lower bits of LFP data seem to be quite noisy. Another possibility would be to extract multiple bits from the sampled LFP data but this would require to evaluate their joint probability distribution. We then computed the XOR of three consecutive bits using a simple two-stage parity filter to increase the entropy density. Figure 5 shows the histogram of 8 consecutive bits after applying the two-stage parity filter. Based on our results, these bits are to some extend uniformly distributed, which demonstrate the potential of the LFP as a randomness source for deriving a symmetric key. Following the test suites from NIST 800-90B [42], we estimated that the Shannon-entropy is around $0.91/bit$. The Shannon-entropy could be further improved by increasing the number of stages of the parity filter.

[4]Using data from mice to extract preliminary results to better understand the human brain is common practice in many scientific disciplines.

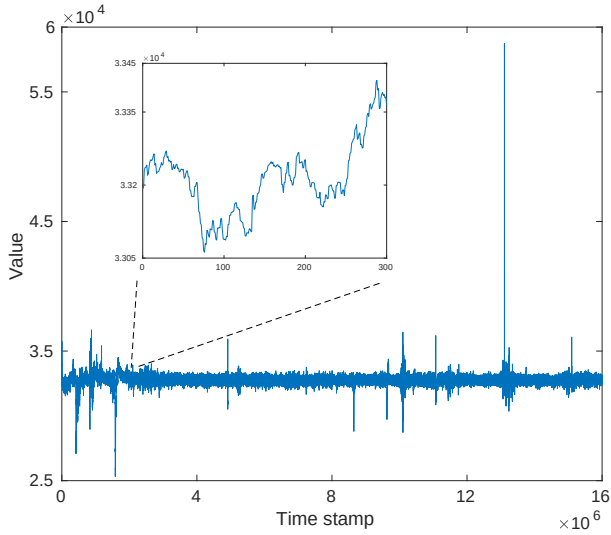

Figure 4: LFP data of one of our mice collected from one channel and sampled with 16-bit precision.

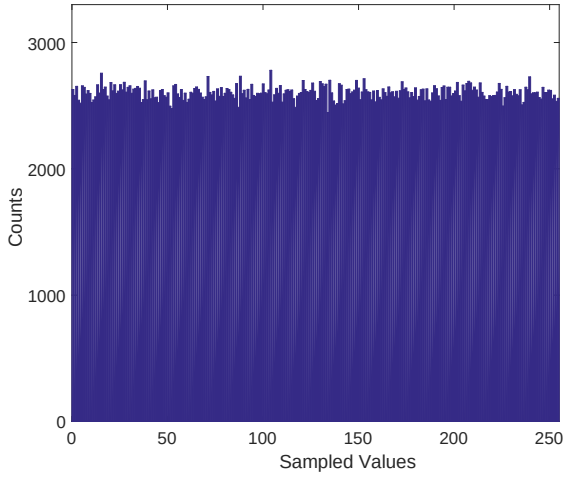

Figure 5: Histogram of bits (in groups of 8 consecutive bits) after applying the parity filter.

7.4 Key Transport

Once a fresh cryptographic session key is generated in the neurostimulator, it has to be securely transported to the device programmer. Our technique for key transportation leverages the fact that both the patient's skin and the neurostimulator's case are conductive. We propose to apply an electrical signal (with the key bits embedded in it) to the neurostimulator's case such that the device programmer can measure this signal by touching the patient's skin, as shown in Fig. 6. To realize this technique, two extra short wires are needed from the microcontroller to the case of the neurostimulator. We discussed this technique with doctors and they claim that applying a signal of a few microvolts or millivolts to the neurostimulator's case would not cause any problem or be unpleasant to patients.

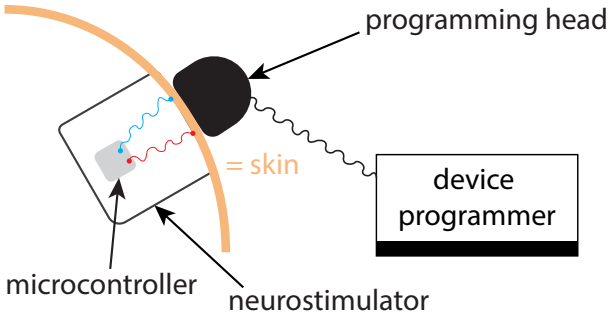

Figure 6: Technique to transport a session key from the neurostimulator to the device programmer.

To evaluate the suitability of this technique, we implemented a proof-of-concept using a NI USB-6351 DAQ and a neurostimulator. To emulate the human body, we used a 1 cm layer of bacon on a 4 cm layer of beef, as suggested by Kim et al. [80]. While this model does not account for changes in the skin conductance, for example, due to patient's emotions or sweat, all these factors can only make the patient's skin to be slightly more conductive than usual. In other words, none of these factors will affect the reliability of our technique. However, this could make it easier for adversaries to capture the EM radiations generated when transporting the key.

To preclude potential EM eavesdropping attacks, our system should be designed to minimize undesired EM radiation produced by the hardware components. This can be done by lowering the transmission power. While this comes at the cost of having a lower data rate, this would not be a problem in our solution. Another way of decreasing the EM emanations would be to use wires (from the

microcontroller to the neurostimulator's case) that are sufficiently short and twisted. To further decrease the EM emanations, one could follow widely used electromagnetic compatibility (EMC) guidelines [4]. In addition, the undesired EM emanations generated from other devices that are in close proximity with the patient (e.g. his smart-phone) could be used to masquerade the EM radiations.

To simulate the process of transporting the key, we created a transmitter and a receiver LabVIEW program. We modulated the data using a standard OOK and set the symbol rate and the sampling rate to 100 symbols/s and 500 ksamples/s, respectively. Given that the duration of each symbol is 10 ms, the 128-bit key can be transported in less than 1.5 s. A few additional delays could be deliberately introduced in the key transport process such that access to the neurostimulator is guaranteed only to device programmers that can touch the patient's skin long enough.

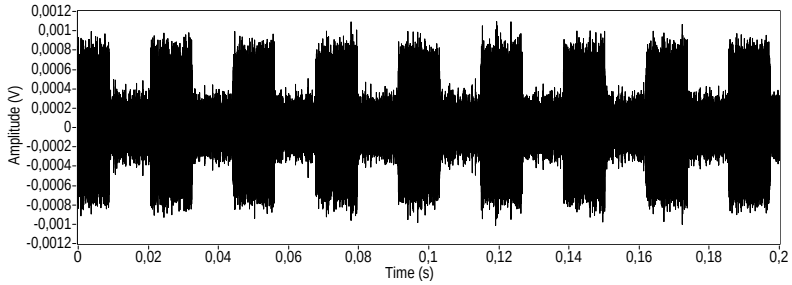

Figure 7: Waveform of the signal received by our DAQ at the other side of the meat.

For our experiments, we connected two wires from the transmission port of the DAQ to the neurostimulator's case. We then placed the meat on top of the neurostimulator, and attempted to measure the transmitted signal at the other side of the meat. Then we mimicked the behavior of a legitimate device programmer by using two wires to touch the meat while connected to the receiver port of the DAQ. The first step was to modulate the "1s" and "0s" corresponding to the 128-bit session key using the OOK modulator. Subsequently, we applied the modulated electrical signal to the neurostimulator's case using our transmitter. A short preamble sequence could be sent before the key is transmitted to help to synchronize the devices. Existing techniques to detect and correct bit errors could also be used to increase the robustness of our technique. Using our receiver, we measured and demodulated the signal to retrieve the key bits previously transmitted by the neurostimulator.

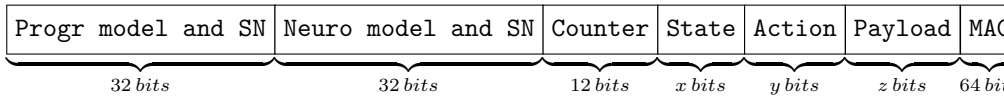

Figure 8: Optimized message format.

We conducted a series of experiments to determine the minimum signal power from which the device programmer can recover the key bits transmitted by the neurostimulator. It is important to note that the transmission power could be even further reduced by using an enhanced multi-feature OOK demodulation scheme, as the one proposed by Kim et al. [80]. Figure 7 illustrates the waveform of the signal (with the key bits embedded in it) that is received by our DAQ at the other side of the meat. This figure shows that it is possible to recover the key bits, which consists of a series of alternating '1s' and '0s', by demodulating a signal whose amplitude is less than 1 mV. Our results show that, when using the parameters mentioned above, the key can successfully be transported from the neurostimulator to the device programmer.

Furthermore, we tested whether it is possible to recover the key without needing to touch the patient's body when using the minimum signal power that we used in the experiment discussed above. For this, we connected two wires to the receiver port of the DAQ and used them as an antenna to try to measure the EM emanations. We repeated this experiment from several distances ranging from 3 meters to a few centimeters. In none of these cases, we were able to capture the transmitted key. This led us to conclude that the key can only be successfully received when touching the patient's skin.

7.5 Secure Data Exchange

For the sake of completeness, we describe how to establish a secure communication channel once the session key has been transported from the neurostimulator to the device programmer. First, both devices could (optionally) execute any authentication protocol to prove to each other knowledge of the shared session key. For efficiency reasons, we chose not to use an authentication protocol. Instead, both devices can implicitly demonstrate knowledge of the key during the first communication session.

The use of cryptography allows for secure data exchange between devices. However, it also increases the energy consumption in both the neurostimulator and the device programmer. This energy consumption can be divided into

two components: (i) computation and (ii) communication cost. The former indicates the cost of performing cryptographic operations while the latter refers to the cost of transmitting/receiving bits to/from a device. However, several papers have shown that the computation cost is often negligible compared to the communication cost [92, 124]. Thus, we propose to use a new optimized message format that is slightly different from the original message format (see Fig. 3). This allows to build security mechanisms into the devices while keeping the additional energy consumption as low as possible in the neurostimulator.

Figure 8 shows the new optimized message format. It includes the same fields as those in the original message except for the two 16-bit checksums, and additionally has a 12-bit counter and a 64-bit MAC that is computed over the entire message. The reasoning behind the proposed message format optimization is the following. A 64-bit MAC offers a good trade-off between cost and security against both off-line and on-line attacks. To prevent replay attacks, the 12-bit counter is increased by one in each message and reset every time a new session key is generated. A message is accepted only if its counter is greater than the one in the previous received message. All these message optimizations lead to an increase of the message size of only 44 bits, which corresponds to an extra communication overhead of less than 10% compared to the original message format.

Since we need to encrypt all message fields except for the SN and model fields of both the device programmer and the neurostimulator, we recommend to use any secure authenticated encryption scheme [3].

8 Limitations and Future Work

Lack of stochastic model for LFP. Our results indicate that the use of the LFP signal is a promising way for extracting randomness in neurostimulators. However, we conducted all our experiments without having a stochastic model on the entropy source or a physical model on the mechanism of the signal origin. In future work, we will develop a stochastic model for the LFP, and conduct a thorough analysis to gather more evidence that LFP can be used as a source of randomness in neurostimulators.

Consider a more powerful adversarial model. Our current solution assumes that adversaries cannot compromise any device that can make physical contact to the patient (e.g. a smart watch). In future work, we want to relax this requirement and also allow the adversary to compromise any wearable device of the patient.

9 Conclusions

In this work we have evaluated the security and privacy properties of a widely used commercial neurostimulator. For this, we fully reverse engineered the proprietary protocol between the device programmer and the neurostimulator over a short-range communication channel. We demonstrated that reverse engineering was possible without needing to have physical access to the devices by using a black-box approach. This allowed us not only to document the message format and the protocol state-machine, but also to discover that the messages exchanged between the devices are neither encrypted nor authenticated. We conducted several software radio-based attacks that could endanger the patients' safety or compromise their privacy, and showed that these attacks can be performed using inexpensive hardware devices. The main lesson to be learned is that security-through-obscurity is always a dangerous design approach that often conceals insecure designs. IMD manufacturers should migrate from weak closed proprietary solutions to open and thoroughly evaluated security solutions and use them according to the guidelines.

To preclude the above attacks, we presented a practical and complete security architecture through which the device programmer and the neurostimulator can agree on a session key that allows to bootstrap a secure communication channel. Our solution grants access to the neurostimulator to any device programmer that can touch the patient's skin long enough. This allows to create a secure data exchange between devices while ensuring that medical personnel can have immediate access to the neurostimulator in emergencies. Our solution accounts for the unique constraints and functional requirements of medical devices, requires only minor hardware changes in the devices and provides backward and forward security.

10 Acknowledgements

The authors would like to thank the anonymous reviewers for their helpful comments and Stefanos Georgoutsos for his support. This work was supported in part by the Research Council KU Leuven: C16/15/058 and by FWO through SBO SPITE S002417N.

A Appendices

A.1 Design of an Antenna Operating at 175 kHz

Since there are no off-the-shelf antennas suitable for transmitting at 175 kHz, this component had to be designed and manufactured in house. The low working frequency points at a loop antenna as the optimal solution. A small loop antenna with N turns and a surface area S carrying an electric current I_o behaves similarly to a magnetic dipole with magnetic moment $I_m l$ given in Eqn. 1:

$$I_m l \approx NSI_o\,. \tag{1}$$

Thus, to maximize the magnetic field component, the equivalent dipole moment should be maximized. The conventional topology for near field communication systems assumes that the antenna size is comparable with the size of the neurostimulator antenna. Thus, the surface area S is determined by the size of the neurostimulator. The number of turns N can easily be adjusted. The electric current I_o depends on the output power and on the antenna matching. A small loop antenna behaves like an inductor with very small losses [41]. So it is not matched with a conventional 50 Ohm output and special antenna tuning is required. Due to the low frequency this tuning circuit can be constructed using lumped elements. The simplest type of matching network is a so called L matching network based on two reactive elements to match almost any load impedance to a 50 Ohm output at a single frequency [59]. Unfortunately, due to the large detuning of the antenna, the antenna impedance and the exact circuit model of the lumped elements, thus including parasitic components, should be known at the working frequency with a very high accuracy. Because the capacitor equivalent series resistance (ESR) can be comparable with the very small antenna resistance, the design of such a small loop antenna is typically based on a trial and error approach. We manufactured the loop antenna from a 4 m long blue 7 x 0.25 mm cable with PVC insulation from LappKabel. The antenna input impedance was measured using a low cost vector analyzer called miniVNA. The measured value was about 1.8 + j 79.4 Ohm at 175 kHz. The input antenna resistance is determined mainly by the Ohmic wire resistance. A perfect matching can be obtained with two ideal capacitors of 2.4 nF and 9.4 nF. Since there were no such capacitors available, the actual tuning was performed by the consecutive testing of different capacitors. The final matching circuit contains two capacitors of 1 nF in parallel and 1 capacitor of 10 nF. All capacitors were fixed on a breadboard to ensure the possibility to easily retune if necessary.

The magnetic field level was tested using a 6 cm H-field probe, Model 901 from the EMCO 7405 Near Field probe kit, and an Anritsu MS2721A spectrum analyzer. At first the field level of an original RF transmission was measured and recorded. Then the recorded message was re-transmitted using the set-up based on our antenna, and the magnetic field level was again measured. The new transmitter with the designed antenna provides a magnetic field level and a bandwidth comparable with those generated by the original neurostimulator.

Publication

A Critical Evaluation of Security Solutions based on Physiological Signals

Publication Data

Eduard Marin, Enrique Argones Rúa, Dave Singelée and Bart Preneel, "A Critical Evaluation of Security Solutions based on Physiological Signals," Currently under submission, 2018.

Contributions

- Principal author. The security design guidelines on how to use physiological signals in security are the result of several discussions with co-authors.

A Critical Evaluation of Security Solutions based on Physiological Signals

Eduard Marin, Enrique Argones Rúa, and Dave Singelée Bart Preneel

ESAT-COSIC and imec, KU Leuven, Belgium

Abstract. A growing number of implantable medical devices (IMDs) are equipped with a wireless interface that enables non-invasive monitoring and reprogramming through an external device programmer. However, the wireless interface also poses major security and privacy risks to patients. The use of cryptography is challenging due to the IMDs' constraints and the inherent tensions between functional requirements, such as availability and utility, and the desirable security and privacy goals. A well-studied problem yet unsolved is how to securely establish and manage cryptographic keys between the device programmer and the IMD. This paper presents a critical evaluation of cryptographic solutions that rely on using patient's physiological signals for the device programmer and the IMD to agree on a cryptographic key. We first challenge several widely adopted assumptions made by cryptographic solutions, and show that they do not always hold in practice. We go on by conducting a thorough analysis of two well-conceived cryptographic solutions, namely the H2H protocol and the Biosec protocol. We found that these solutions have serious security weaknesses and design flaws. Finally, we describe the process of how to securely use patient's physiological signals in cryptographic solutions, and discuss research directions for future work.

1 Introduction

Implantable medical devices (IMDs) such as pacemakers, implantable cardioverter defibrillators (ICDs) or insulin pumps, are used to monitor and treat patients with chronic diseases such as arrhythmia or Parkinson disease. Currently, the vast majority of IMDs is equipped with a radio interface that allows wireless communication with external devices, also known as device programmers. Device programmers enable doctors not only to gather telemetry

data and patient information stored in the IMD, but also to modify the IMD's settings remotely. This helps to enhance the patients' life quality by providing more flexible and personalized treatments while improving the sustainability of health-care systems. However, despite the benefits of the wireless interface, there are also significant security concerns for patients. For example, adversaries could eavesdrop the wireless channel to compromise the patient's privacy, or send malicious commands to the IMD to alter its settings or disable the therapy. The latter can endanger the patient's life and in worst case can lead to fatal consequences for patients.

Several articles have already shown that IMD manufacturers often rely on hiding the protocol specifications as a means to provide security (i.e. security-through-obscurity). Moreover, an important body of research has identified several security shortcomings in these systems. For example, Hei et al. introduced simple yet effective denial-of-service (DoS) attacks against IMDs that cannot be prevented with traditional cryptographic solutions [73]. These attacks can reduce the IMD's battery lifetime from several years to a few weeks only. Halperin et al. analyzed the proprietary wireless communication protocol between a device programmer and an ICD to communicate over a short-range communication channel (less than 10 cm) [70]. As no security mechanisms were found, they were able to perform various attacks, that range from changing the therapy to inducing fibrillation, by just replaying past transmissions sent by legitimate device programmers. Marin et al. investigated the security of the proprietary protocol between a device programmer and the latest model of an ICD over a long-range communication (few meters) [91]. After fully reverse engineering the protocol using a black-box approach, they were able to conduct several types of attacks using only inexpensive hardware devices. Similarly, Li et al. reverse engineered the proprietary protocol between a remote control and an insulin pump [86]. Marin et al. extended the previous work by reverse engineering the proprietary protocol between an insulin pump and all its potential peripherals [92].

Prior work has also studied countermeasures to protect the wireless communication channel between the device programmer and the IMD. Among these countermeasures, a promising approach for the device programmer and the IMD to establish a cryptographic key is to use a physiological signal of the patient, such as the electrocardiogram (ECG), as a source of randomness.

Contributions: This paper provides a critical evaluation of key establishment and authentication protocols that rely on physiological signals. In detail, our main contributions can be summarized as follows:

- **Identification of unrealistic assumptions.** We challenge several assumptions that are widely adopted, and explain why they do not hold in real scenarios.

- **Attacks against two well-known cryptographic solutions.** We discovered several protocol weaknesses and design flaws in the H2H and Biosec.
- **Propose a set of security design guidelines.** We describe all the steps that need to be followed in order to securely use patient's physiological signals in cryptographic solutions.

Paper outline: The remainder of this paper is organized as follows. Section 2 gives an overview of related work. Section 3 challenges several widely used assumptions considered in previous works and demonstrates that some of them do not hold in real scenarios. Section 4 shows that the H2H protocol and the Biosec protocol have serious protocol weaknesses and design flaws that make them vulnerable to attacks, whereas a set of security design guidelines that describe the process of how to securely use physiological signals in cryptographic solutions is given in Section 5. Section 6 discusses several research directions for future work. Finally, Section 7 gives concluding remarks.

2 Related Work

Several types of security solutions have been proposed to create a secure data exchange between the device programmer and the IMD. These countermeasures can be grouped into three main categories: (i) using external devices as shields, (ii) using auxiliary or Out-Of-Band (OOB) channels and (iii) using patient's physiological signals. In this section, we review each of these solutions.

2.1 External Devices

Gollakota et al. proposed to use an external device, also known as "shield", in order to mediate between device programmers and legacy IMDs. It relies on *friendly jamming* to jam the messages to/from the IMD in order to prevent unauthorized entities from decoding them, while still being able to receive and decode them itself. While the shield can alleviate some security issues, it also has important downsides. Specifically, the shield can interfere with ongoing communications between legitimate devices, and does not protect against high-powered adversaries. Furthermore, the shield provides only weak confidentiality guarantees since multiple input multiple output (MIMO) adversaries can cancel out the jamming signal and recover the message content, as shown by Tippenhauer et al. [128]. Xu et al. presented a wearable device called "IMDGuard" that authenticates the device programmer on behalf of the ICD

through an ECG-based key establishment protocol [134]. However, Rostami et al. found that the IMDGuard is vulnerable to a Man-in-The-Middle (MiTM) attack which reduces its effective key length (EKL) from 129 bits to 86 bits [113]. Zhang et al. introduced "Medmon", a multi-layered anomaly detection system that relies on physical and behavioral anomaly detection to detect malicious actions [136]. Depending on the severity of the threat, Medmon can either warn the patient or jam the wireless channel to block malicious messages. Yet, Medmon is energy-consuming, offers very limited security protection and is neither flexible nor user-friendly. Another limitation of all these approaches is that jammers are illegal in many countries and their use can result in large fines.

2.2 Auxiliary or OOB Channels

Halperin et al. presented a zero-power authentication protocol by which the device programmer and the IMD can agree on a key. For this purpose, a random symmetric key is first generated in the IMD and then transported to the device programmer over an audio channel [70]. The authors stated that the key can only be picked up by device programmers that touch the patient's skin. However, Halevi et al. demonstrated that it is possible to recover the key by eavesdropping the acoustic channel from a few meters away [68]. Kim et al. [80] and Abhishek Anand et al. [38] proposed two similar pairing schemes that consist of transmitting a key over a vibration channel while jamming the acoustic channel to mask the acoustic emanations produced as a side effect of the vibration. Although this technique prevents adversaries from recovering the key transmitted over the acoustic channel, it remains unclear what the effects are on the key sent over the vibration channel, and also whether the vibration channel can be eavesdropped. Rasmussen et al. proposed an access control system that is based on using ultrasonic distance bounding in combination with the Diffie-Hellman protocol [109]. In this solution, the IMD establishes a key with any device programmer that is in its close proximity, meaning that this provides security only under the assumption that the adversary cannot be sufficiently close to the patient.

2.3 Using Patient Physiological Signals

The use of physiological signals (PS) extracted from the patient for generating and establishing a cryptographic key between two (or more) devices was first proposed by Poon et al. [104]. In contrast to biometrics, which are to some extend time-invariant, PSs are random signals that vary over time. Examples

of PSs used in authentication and key establishment protocols include the ECG, photoplethysmogram (PPG), blood glucose, blood pressure, temperature, hemoglobin and blood flow [135]. For example, Poon et al. [104] and Rostami et al. [114] showed that the time between patient heartbeats, also known as interpulse interval (IPI), exhibits two desirable properties: (i) it provides a high level of randomness and (ii) it can be measured anywhere in the patient's body by touching her skin.

A common approach for establishing a cryptographic key between the device programmer and the IMD is that both devices take a measurement of the chosen PS independently and synchronously [90]. These PS measurements are typically not equal and at best only rather similar due to the noise, and not necessarily uniformly distributed. However, cryptographic keys need to be random, uniformly distributed and identical on both sides of the communication. The problem of how to generate cryptographic keys from noisy data was first studied by Dodis et al. [57]. Prior work in the domain of medical device security has proposed to use PSs in combination not only with fuzzy cryptographic primitives, as suggested by Dodis et al., but also with cryptographic commitment schemes. Below, we give an overview of the existing solutions belonging to each of these groups.

1) Fuzzy cryptographic primitives: Miao et al. proposed to modify the original fuzzy vault so that it can be applied in the context of body sensor networks [96]. K Venkatasubramanian et al. presented a key agreement protocol based on the use of a fuzzy vault, also known PKA, which uses PPG signals as a means for sensors to agree on a cryptographic key [131]. K Venkatasubramanian et al. also proposed PSKA, a key agreement protocol that uses a fuzzy vault to allow sensors to securely communicate with each other without requiring any initialization phase or pre-deployment of keys [132]. Yet, Bagade et al. showed that it is possible to break PSKA with an average probability of 30% after a 30-second handshake [40]. Furthermore, PSKA does not provide high security guarantees due to the limitations on the feature size, and requires a complex algorithm to generate the chaff (i.e. random) points. Intuitively, one could increase the number of chaff points to enhance the security of PSKA. Nevertheless, this would result in collisions between the chaff points and the legitimate points generated by the other sensor. Hu et al. proposed a fuzzy-vault-based key agreement protocol – called OPFKA – which overcomes the limitations of PSKA [77]. However, Rostami et al. showed that the OPFKA is vulnerable to an attack that exploits the use of a hash function to expand the feature size [113]. Cherukuri et al. proposed a key distribution protocol, also known as Biosec, wherein PSs extracted from the patient's body are used to securely transport a session key between two sensors [49]. Biosec relies on a fuzzy commitment scheme and enables sensor re-keying.

2) Commitment scheme: Rostami et al. presented heart-to-heart (H2H), a commitment-scheme-based pairing protocol through which the device programmer authenticates to the IMD without needing to share any prior secrets [114]. H2H ensures access to the IMD by any device programmer that can touch the patient's skin to measure her IPI. For this, both the device programmer and the IMD need to take a measurement of the patient's IPI independently and synchronously. These measurements are then used as a "fuzzy password" that is known only to them.

In the next sections, we will discuss several limitations and security problems in the H2H and Biosec protocols.

3 Unrealistic Assumptions and Adversarial Models

In this section, we challenge several widely adopted assumptions when deploying security solutions in IMDs, and show how they can be bypassed in practice.

- ***The device programmer will only be used by medical professionals.***

  Currently, most device programmers are vendor-specific and lack security measures to authenticate the doctor, meaning that anyone that possesses a device programmer can interact with all IMD's models of a specific manufacturer. This introduces important security risks for patients with an IMD since adversaries can easily purchase a device programmer on the Internet[1] or steal it from a hospital. Indeed, according to the 2014 Healthcare Breach Report from Bitglass [21], many hospitals struggle to keep their devices and equipment away from adversaries. Surprisingly, this report shows that there are more data breaches due to loss or theft of devices than to security attacks by hackers.

- ***It is difficult for adversaries to obtain a master key.***

  In the literature, there are several security solutions that use a master key to bootstrap a secure communication between the device programmer and the IMD (e.g. [70]).

  As previously mentioned, adversaries can purchase or steal a device programmer to mount attacks against patients with an IMD. If this approach is used, there is no need for extracting the master key. However, this approach has some limitations since device programmers are often

[1] By the time we wrote this paper, there were several device programmers from different manufacturers on sale in eBay.

big and heavy and hence do not fit in a backpack. Instead, adversaries can attempt to extract the master key from a device programmer located in a hospital through side-channel or physical attacks (e.g. [82], [37]). Once the key is compromised, they can use their own commercial off-the-shelf (COTS) hardware devices to conduct attacks. In contrast to device programmers, these devices are more inexpensive and portable. While these attacks are rather straightforward if no security measures are deployed, there are ways to increase the difficulty of performing them. For example, by storing the master key in secure hardware on the device's programmer.

In addition to securely storing the master key in the device programmer, other security mechanisms should be enforced to avoid breaches in the supply chain that expose the key to adversaries. For example, the master key should be kept in a server that is physically protected (e.g. put in a secure place where it cannot be stolen or tampered with). In addition, an adequate access control system should be implemented so that only authorized employees have access to the server through appropriate identification/authentication mechanisms. But even if all these security measures are deployed, the master key could still be leaked through insider adversaries.

The previous examples show that the attack surface is quite large. If adversaries ever manage to obtain the master key using any of these methods, they could compromise the security of millions of devices. This poses a serious threat to solutions that are based only on factory-installed secrets stored in device programmers.

- ***Adversaries cannot make physical contact or be sufficiently close to the patient*** i.e. physical contact or proximity between the device programmer and the IMD enables private and/or authentic communications.

  Several solutions have been proposed that use electromagnetic, vibration or acoustic OOB channels as a means to authenticate or establish a key between the device programmer and the IMD [109,114]. The security of all these solutions relies on the assumption that adversaries cannot eavesdrop or tamper with the data sent over these channels. However, as shown in Section 2, all these solutions are vulnerable to remote eavesdropping attacks (e.g. [68, 86]). These attacks make it possible for adversaries to intercept the data transmitted over these OOB channels without needing to be nearby. Nevertheless, adversaries still need to make physical contact or be in close proximity with the patient in order to perform attacks. This is because device programmers typically activate (or interact with) IMDs over a short-range channel that requires the device programmer

to be located a few centimeters away from the IMD. However, there are scenarios, such as a crowded subway, where adversaries can be in close proximity with patients to execute these attacks.

- ***PS cannot be obtained remotely***, i.e. PSs can only be picked up reliably when the device programmer makes physical contact with the patient.

  Prior work has demonstrated that some PSs, such as those extracted from patient's heart, can be gathered without the need for being in close proximity with the patient. This can be achieved using several methods. For example, Calleja et al. showed how to collect information of cardiac signals remotely using inexpensive hardware equipment [47]. Seepers et al. showed that well-known remote photoplethysmography (rPPG) techniques allow to measure the heart rate with similar accuracy as when doing so by touching the patient's skin [122]. Poh et al. presented a technique through which the patient's heart rate can be measured using a standard webcam [103]. Katabi et al. showed that even WiFi signals can be used to infer the breathing and heart rate of individuals [24]. This renders all security solutions based on these PSs insecure, and shows that PSs extracted from the patient's heart are particularly vulnerable to attacks.

4 Security Attacks and Design Flaws in PS-based Security Solutions

In this section, we show that both the Biosec and the H2H have serious protocol weaknesses and design flaws which render them insecure.

4.1 Attacks on H2H

By using public-key cryptography in combination with a commitment scheme, the H2H protocol enables the IMD and the device programmer to securely exchange data. H2H implements a novel access-control policy called "touch-to-access" which ensures access to the IMD by any device programmer that can make physical contact with the patient to measure her heart rate. Figure. 1 provides an overview of the H2H pairing protocol. H2H can be divided into two different phases: (i) *a secure-channel setup phase* and (ii) *an authentication phase*.

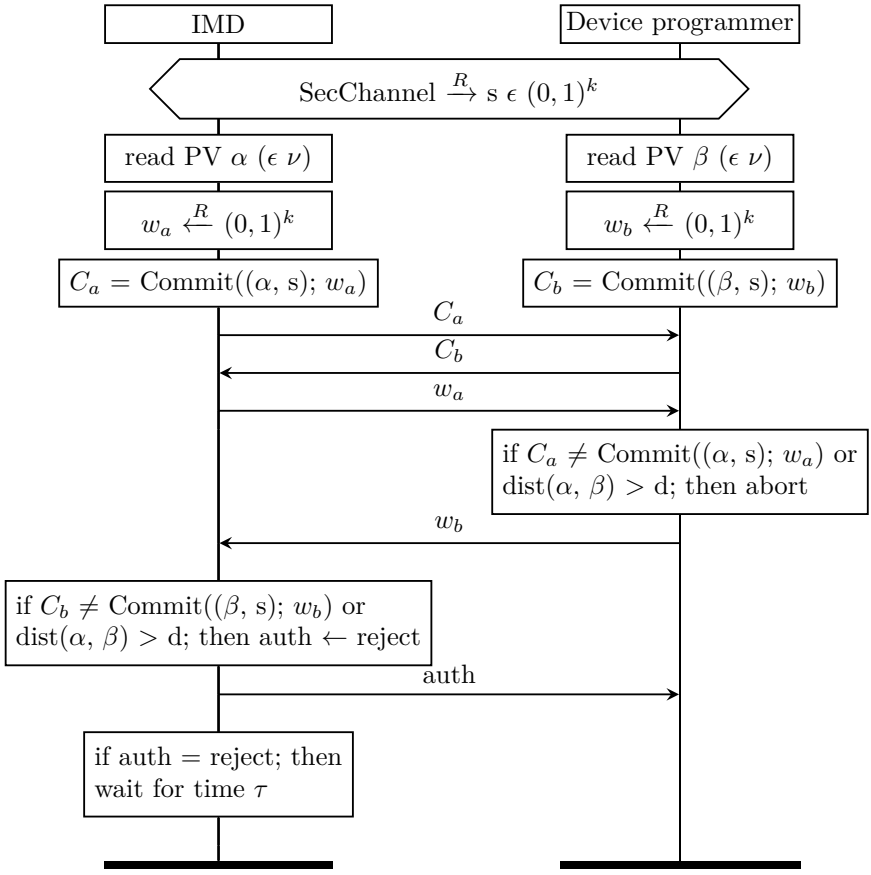

Figure 1: H2H pairing protocol proposed by Rostami et al.

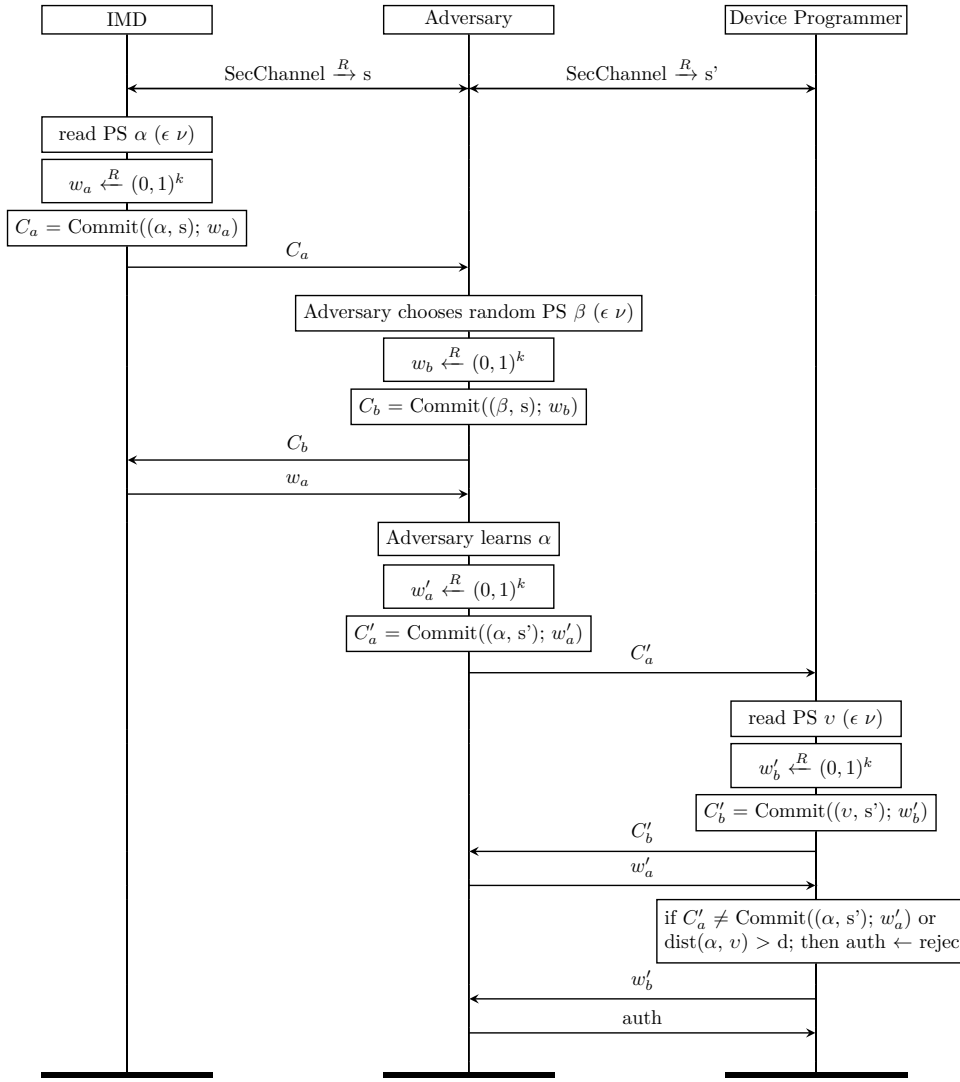

Figure 2: MiTM attack. The adversary wants to trick the device programmer into believing that it is communicating with the legitimate IMD while it actually does it with him.

In the *secure-channel setup phase*, a secure but unauthenticated channel is first created between the IMD and the device programmer using the transport layer security (TLS) protocol. This channel provides confidentiality, integrity and freshness. The IMD takes the role of a TLS client whereas the device programmer acts as a TLS server. The device programmer presents its certificate to the IMD but the latter does not verify it. This is to avoid the burden of needing a public-key infrastructure (PKI). The secure-channel setup phase outputs a unique, random number s which does not need to be kept secret, e.g. the hash of the TLS master key and the public key.

In the *authentication phase*, the IMD authenticates the output value of the previous phase s by using the touch-to-access policy described above. This is achieved using a commitment scheme that allows both devices to commit to their respective IPI measurements in a secure manner.

The first step for the IMD and the device programmer is to take an IPI measurement denoted by α and β, and generate a random number w_a and w_b, respectively. Each of the devices binds its commitment (C_a and C_b) to s in order to avoid that adversaries reuse α or β on a different channel. Once C_a and C_b have been exchanged, the IMD sends w_a to open the commitment C_a, allowing the device programmer to obtain α. Subsequently, the device programmer verifies whether α, w_a and s were used to produce C_a, and that the distance between α and β is less than a predetermined threshold according to some metric. If these conditions are met, the device programmer sends w_b to open the commitment C_b, allowing the IMD to obtain β. Similarly, as before, the IMD verifies whether these parameters were used to produce C_b, and that the distance between α and β is within a predefined threshold. If these conditions are satisfied, the IMD considers the public key included in the certificate it received during the secure-channel setup phase as authentic.

We want to emphasize that the H2H lacks important details about both the TLS protocol that is run in the secure-channel setup phase and the commitment scheme that is used in the authentication phase. Without these details, it is difficult to assess the security offered by this protocol. The security of H2H is based on the assumption that the IPI measurements (α and β) are secret one-time values that are only known to the IMD and the device programmer, which can be revealed at the end of each protocol instance. However, we show that it is possible to conduct two attacks against H2H, namely a reflection and a MiTM attack, without needing to touch the patient's skin to learn the patient's IPI.

Reflection attack: We found a simple yet effective attack where adversaries can gain access to the IMD without needing to know the patient's IPI. Our attack leverages the fact that the H2H protocol is symmetric in both directions (i.e.

from the IMD to the device programmer and vice versa), and works as follows. The adversary first executes the secure-channel setup phase to establish a TLS session with the IMD. Recall that the IMD does not validate the certificate it receives from the device programmer. In the authentication phase, the goal of the adversary is to authenticate s (the TLS output) without having access to any IPI measurement. However, we observed that adversaries can simply replay the messages the IMD sends. In other words, the adversary can choose C_b and w_b to be equal to C_a and w_a. This tricks the IMD into believing that it executes the TLS protocol with a device programmer that can measure the patient's IPI. To fix this flaw, the IMD can simply reject C_b if it is identical to C_a. Another possible solution would be to modify the protocol such that it is no longer symmetric.

MiTM attack: The H2H protocol is susceptible to a MiTM attack where adversaries convince the device programmer that it is communicating with the legitimate IMD while it actually does it with a fake IMD (i.e. the adversary).

The attack is shown in Figure. 2 and works as follows. The adversary first executes the secure-channel setup phase with the IMD and the device programmer, respectively, to establish two simultaneous independent TLS sessions s and s'. In the authentication phase, the IMD creates its own commitment C_a and sends it to the device programmer (in that case the adversary). Upon receiving C_a, the adversary creates its own commitment C_b and sends it to the IMD. Both C_a and C_b are binded to s. C_b can be whatever random value the adversary chooses. The next step for the IMD is to send w_a to allow the opening of the commitment C_a, which enables the adversary to obtain α (the IPI measurement taken by the IMD). As soon as the adversary learns α, he aborts the session with the IMD, creates a new commitment C'_a that contains α, and sends it to the device programmer. The device programmer then creates and sends its commitment C'_b. Both C'_a and C'_b are sent over s'. After receiving C'_b, the adversary sends w'_a, which allows the device programmer to open the commitment C'_a. Following the same procedure, the device programmer sends w'_b such that the IMD (in that case the adversary) can open the commitment C'_b. Finally, the adversary can simply reply with an *auth* message without needing to open the commitment C'_b. Upon receiving *auth*, the device programmer is convinced that it is communicating with the IMD over s, but actually it is doing it with the adversary over s'.

4.2 Security Analysis of Biosec

The goal of the Biosec protocol is to use PSs extracted from the patient's body to securely exchange data between two sensors. Fig. 3 provides an overview of

the protocol proposed by Cherukuri et al. Before outlining how the protocol works, we give an overview of the variables and cryptographic primitives used throughout the protocol.

Notation: We denote by *Data*, *eData* the data in an unencrypted and encrypted form, respectively. m corresponds to a 128-bit MAC tag whereas K_s is a 128-bit key generated by S_1 and transported to S_2. m_s is a 128-bit random number generated through various PSs whereas r_u is a 128-bit static patient's ID. K_c is a 128-bit sequence that is computed by XORing m_s and r_u. Its function is to mask K_s. *Scom* is a 128-bit sequence that is the result of masking K_s with K_c. We denote the same variables as m'_s, K'_c, K'_s when they are generated/computed by S_2. Furthermore, Biosec assume that all sensors use an error correction code (ECC) that can correct up to T errors. For this, the following equation needs to be satisfied: T = (D - 1)/2; where D is the minimum distance of the ECC. H denotes a one-way function (e.g. a hash function).

The Biosec protocol works as follows: S_1 generates a random session key K_s and encrypts some information *Data*. This results in a ciphertext, *eData*, i.e. $eData = E_{K_s}(Data)$. Subsequently, a MAC tag m is computed over *eData* ($m = MAC_{K_s}(eData)$). To transport the key K_s from S_1 to S_2 without revealing it to adversaries, S_1 first adds some redundancy to K_s (depending on the type of ECC), conceals its value using K_c and computes $H(K_s)$. *eData*, m and *Scom* are then sent to S_2, where $Scom = \mathrm{H}(K_s) \parallel (K_s \oplus K_c)$. Subsequently, S_2 attempts to undo the masking operation in order to obtain K_s from *Scom* through the use of K'_c. Ideally, if the PS measurements were identical, it would be trivial for S_2 to obtain K_s. In practice, though, as m'_s is not equal but rather similar to m_s, K'_s is slightly different than K_s. To retrieve K_s from K'_s, S_2 applies the ECC previously used by S_1, computes $H(K'_s)$, and verifies whether the result corresponds to $H(K_s)$. If this condition is satisfied, K_s can be used to verify m and decrypt *eData*.

The security of Biosec is based on the assumption that adversaries cannot obtain K_s by intercepting the messages sent over the air. However, we demonstrate that this assumption do not always hold. In particular, we want to highlight that the Biosec protocol lacks a rigorous security analysis which lead to several important design flaws.

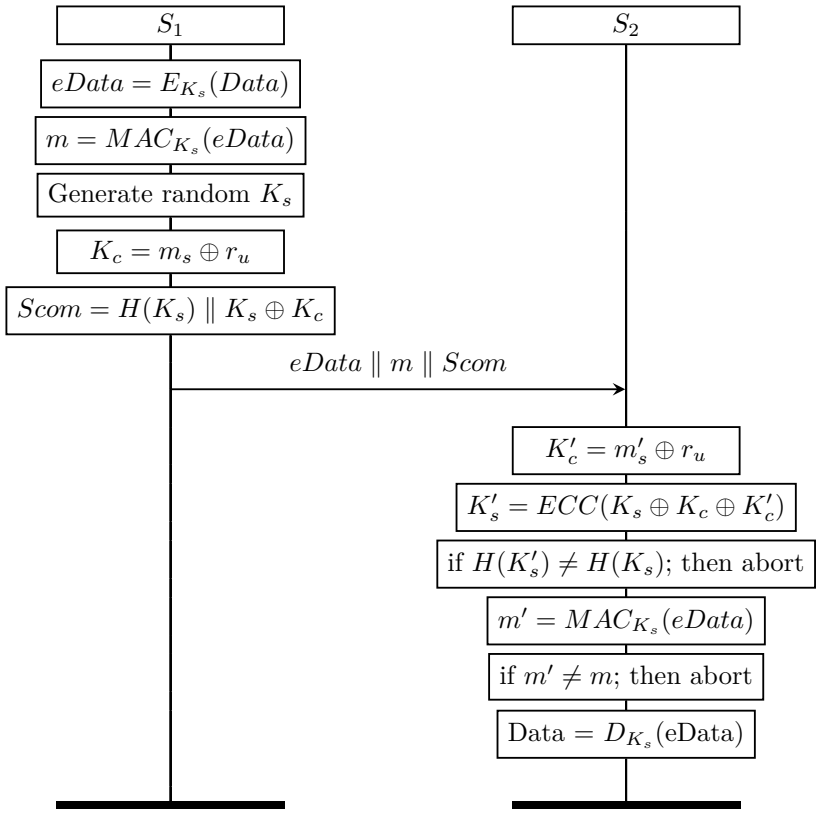

Figure 3: Key distribution protocol proposed by Cherukuri et al.

- **K_c randomness**: As previously explained, K_c is the result of XORing several PSs with r_u. Its function is to conceal K_s while it is being transported, preventing adversaries from obtaining this key while it is transported from S_1 to S_2. However, it is unclear what the purpose is of using the static ID and how S_2 knows r_u. We argue that r_u does not provide any extra security since the masking operation (i.e. XOR) is linear and hence its effect can be cancelled out just by capturing two messages and subtracting their K_c values.
- **PSs randomness**: As it has been pointed out by the authors, another possible shortcoming of Biosec could be the lack of sufficient PSs entropy.
- **Reuse of the key**: We observed that a single key is used for providing

confidentiality and authenticity. Using one key in multiple cryptographic primitives is considered to be "bad practice". An easy fix to this problem would be to use K_s to derive two independent random keys using a standard key derivation function (KDF), or even better, use authenticated encryption.

- **Using an ECC**: The use of an ECC reduces the EKL since it adds redundancy to correct errors, thus reducing its entropy. A possible solution would be to increase the key length to achieve the same level of security that is expected when no ECC is used. The authors, though, define K_s as a 128-bit key and do not consider the loss of entropy. When using an ECC to correct a significant number of errors, the EKL can be reduced considerably. This could make Biosec vulnerable to brute-force attacks where adversaries can try all key combinations. This type of attack can be carried out regardless of the randomness level of the PSs and the type of masking operation being used. For each key attempt, adversaries can check whether they successfully guessed the key since the hash of the key being used is included in the message. In addition, adversaries could potentially perform these computations off-line and create a table with all possible keys and their corresponding hash values. Thus, adversaries only need to capture one message sent from S_1 to S_2 and find the match in the table in order to recover the key.

5 How to Use Physiological Signals in Security?

5.1 Criteria for Selecting Physiological Signals

PSs need to satisfy several properties, some of which are similar to those used in biometric systems.

- ***Readiness***: The process of acquiring the PS should be as fast as possible.
- ***Exclusivity***: Only the device programmer and the IMD should be able to accurately measure the PS.
- ***Availability***: The PS should depend neither on the patient nor on his health condition.
- ***Entropy***: The PS should provide high entropy so that it is possible to generate random keys in short periods of time. More specifically, the PS should offer high *inter-subject* variability and high *intra-subject* variability.

- ***Precision***: The differences between the PS measured by the device programmer and IMD should be as small as possible to maximize the EKL and reduce the acquisition time.

5.2 Fuzzy Extraction from Physiological Signals

The use of PSs for generating, transporting and establishing keys needs to be carefully formulated in order to avoid undesired information disclosures. For this purpose, the most appropriate formal construction is the *fuzzy extractor*, introduced by Dodis et al. [57]. The properties of fuzzy extractors have been widely analyzed and are well understood. Informally, fuzzy extractors comprise the following two primitives:

- ***Generate***. This primitive is a randomized procedure that takes a fuzzy measurement r as an input and outputs a public string P as well as a secret string S. S can be used as a cryptographic key or as a seed in a KDF.
- ***Reproduce***. This primitive consists of two inputs, namely a fuzzy measurement r' and the public string P previously generated by *Generate*. If the distance between r and r' is below a given threshold t, *Reproduce* outputs the same secret string S as the one produced by *Generate*.

In this domain, both the device programmer and the IMD need to independently and simultaneously take a PS measurement, which is denoted by r and r', respectively. Subsequently, one party executes *Generate*, whereas the other performs *Reproduce*. The idea behind fuzzy extractors is that, if r and r' are not equal but similar, both parties will be able to agree on the same cryptographic key. Prior work has also analyzed the entropy loss on the fuzzy measurement due to the disclosure of the public string P, which is also known as helper data [57]. This depends on the PS entropy and the precision, i.e. the similarity between the measurements taken by the device programmer and the IMD.

In addition to all properties that PSs need to satisfy, there are other practical aspects that need to be considered. For example, the use of fuzzy extractors increases the energy consumption in both the device programmer and the IMD. Specifically, the energy consumption can be divided into two main components: the *computation* and the *communication* cost. The former refers the cost of conducting operations such as ECC decoding, whereas the latter refers to the cost of transmitting/receiving bits to/from a device.

Below, we discuss each of these practical aspects more in detail.

Helper data. Beyond the helper data derived by the *Generate* procedure, quantization helper data can also be produced. Zero-leakage quantization helper data does not disclose any information about the discrete PS representation, links the continuous and discrete representations, and reduces the FAR and FRR [53]. In this scenario, the FAR represents the probability that a random PS measurement produces the same secret string as the one produced by *Generate*. Similarly, the FRR corresponds to the probability that a genuine PS measurement produces a different secret string than the one produced by *Generate*.

A well-known drawback of fuzzy extractors is that the use of helper data increases the communication burden and hence the energy consumption in both the device programmer and the IMD. This is because the helper data needs to be transmitted from the party that executes *Generate* to the one that executes *Reproduce*. When using quantization helper data, additional information needs to be transmitted, increasing even more the communication burden. However, using quantization helper data reduces the number of expected errors, and therefore the complexity of *Reproduce*.

Complexity and energy consumption. In contrast to device programmers, which are external powerful devices, IMDs are resource-constrained devices that pose severe limitations on the computation complexity and energy consumption. Thus, it is important to conduct a thorough analysis to estimate the complexity and energy cost required to execute both the *Generate* and *Reproduce* procedures.

The *Generate* procedure is typically less computationally complex than the *Reproduce* procedure. Practical implementations of fuzzy extractors, such as the fuzzy commitment, use a remarkably simple *Generate* procedure, and the complexity is mostly concentrated on the *Reproduce* procedure. Yet, the *Generate* procedure requires some randomness. We argue that the device programmer can easily generate randomness from different entropy sources. However, it is certainly a non-trivial task to extract randomness in resource-constrained devices like IMDs. Furthermore, the party executing the *Generate* procedure has to transmit helper data – which could also include quantization helper data – to the other party. Marin et al. [92] showed that communication cost is dominant compared to the energy cost required to compute low-complexity cryptographic primitives such as symmetric encryption/decryption. In this domain, it is unclear whether the communication cost is still dominant or the ECC-related operations (i.e. computation cost) are more energy hungry

Security and usability considerations. Another important aspect is how to set the working point of the fuzzy primitives. Figure 4 shows the error rates as functions of the EKL for a 255-bit synthetic signal. For this, we assumed constant error rates per bit for the legitimate and adversarial signals, and used BCH ECCs [45, 75]. The EKL is the length of the message encoded by the ECC in a fuzzy commitment scheme. For example, when the EKL is 91 and the code length is 255, the code used is a BCH with length 255, dimension 91, and error correcting capability 25. Ideally, the EKL and the FAR are related by the following equation:

$$\mathrm{FAR} = 2^{-\mathrm{EKL}} \tag{1}$$

However, in practice the most usual case is that $\mathrm{FAR} \gg 2^{-\mathrm{EKL}}$. This is because of an imperfect source coding. In other words, the entropy of the discrete features provided to the fuzzy extractor is far from maximum. The EKL is related to the resilience of a fuzzy extractor to be spoofed by an adversary who does not have access to a genuine PS source. This adversary succeeds with probability $2^{-\mathrm{EKL}}$. Similarly, the FAR is the probability of success of an adversary who samples a genuine PS source.

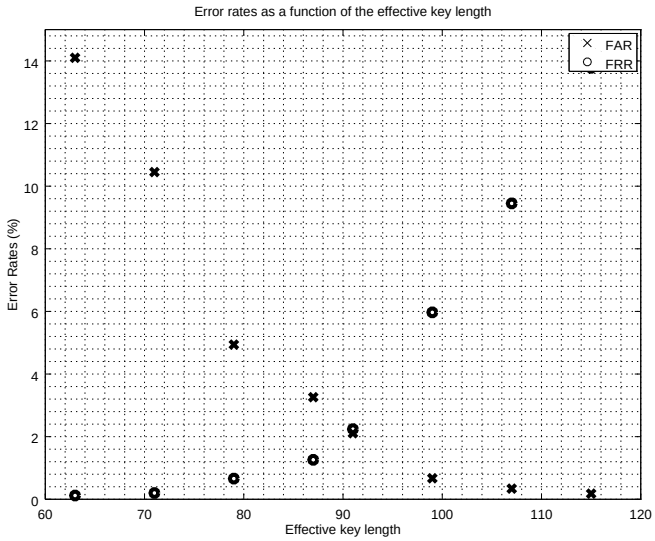

Figure 4: Example of False Acceptance Rate and False Rejection Rate as a function of the EKL for 255 code length BCH codes.

As illustrated in Figure 4, the more error correction capabilities are used, the lower the FRR becomes. This also results in a shorter EKL. FRRs are closely related to availability, since a high FR would make the device pairing less reliable and hence more time consuming. In general, fuzzy primitives provide a variety of available working points and security trade-offs for the same signal. As not all the functions and sessions of a device programmer-IMD might have the same requirements, different working points can be set. Session-specific security and availability requirements can be specified to select the most appropriate ECC and working point.

6 Future Work

In this section, we discuss several potential research directions for future work.

New physiological measurements. New PSs can be investigated that meet the necessary requirements in terms of *Readiness*, *Exclusivity*, *Availability*, *Entropy*, and *Distinctiveness*.

Efficient and flexible fuzzy primitives. To minimize the energy consumption in the IMD due to the use of fuzzy extractors, an analysis is needed to determine which of the two types of energy costs is the dominant. This will determine the role played by each of the devices and the types of ECC that will be employed. At the same time, more research is needed that explores whether and how to design more efficient and flexible fuzzy primitives.

Doctor authentication. As previously mentioned, anyone in possession of a device programmer can use it to interact with an IMD. However, as device programmers are non-constrained external devices, there are several ways to ensure that only legitimate doctors can use device programmers. For example, one possible solution would be to give doctors an authentication token or smart card with cryptographic capabilities containing a public-private key pair signed by the manufacturer.

Hybrid approaches. One possible solution to overcome the limitations of some the existing security solutions would be to combine them in an hybrid approach. For example, using a factory-installed key and a proximity or contact-based secret sharing scheme. In such systems, the factory-installed keys would not be a single point of failure, since the adversary would also need to get an accurate PS in order to obtain to the session key. Equivalently, having access to the PS, or simply eavesdropping the short-range communication would not suffice either, i.e. the adversary would need to simultaneously be in possession of the factory-installed key. However, while an hybrid solution would significantly

increase the security protection, it could also have a negative effect on the IMD's reliability. Thus, more research is needed to investigate how to combine several security solutions without increasing the overall energy cost and significantly decreasing the IMD's reliability.

7 Conclusions

This paper has presented a critical evaluation of security solutions protocols that rely on patient's physiological signals for generating and establishing a cryptographic key between the device programmer and the IMD. We showed that most of these solutions rely on unrealistic assumptions that underestimate the adversaries capabilities. Furthermore, our work revealed serious security weaknesses in two well-known cryptographic solutions, namely the H2H and the Biosec protocol. Finally, we provided a set of recommendations on how to securely use physiological signals in cryptography. The observations and lessons learned from this work can facilitate the process of how to design both more efficient fuzzy cryptographic primitives and secure protocols.

Acknowledgment

The authors would like to thank the anonymous reviewers for their helpful comments. This work was supported in part by the Research Council KU Leuven: C16/15/058.

Bibliography

[1] American Academy of Pain Medicine. http://www.painmed.org/patientcenter/facts_on_pain.aspx.

[2] Binary Ninja. https://binary.ninja/.

[3] CAESAR Competition. https://competitions.cr.yp.to/caesar.html.

[4] EMC Improvement Guidelines. http://www.atmel.com/images/doc4279.pdf.

[5] Federal Communications Commission (FCC) ID. http://www.fcc.gov/encyclopedia/fcc-search-tools.

[6] Federal Communications Commission. MICS Medical Implant Communication Services, FCC 47CFR95.601-95.673 Subpart E/I Rules for MedRadio Services.

[7] LabVIEW. http://www.ni.com/labview.

[8] Medical Devices and EMI: The FDA Perspective. https://www.fda.gov/MedicalDevices/ucm106367.htm.

[9] MPLAB Integrated Development Environment. https://www.microchip.com/mplab/mplab-ide-home.

[10] MSP430FRxx FRAM Ultra-low-power Microcontrollers. http://www.ti.com.

[11] NI USB-6351. http://www.ni.com/en-us/support/model.usb-6351.html.

[12] NI USRP-2920. http://www.ni.com/en-us/support/model.usrp-2920.html.

[13] OpenMSP430 Project. http://www.opencore.org/.

[14] Parkinson Association. http://www.parkinsonassociation.org/facts-about-parkinsons-disease/.

[15] Random Number Generation Using the MSP430. http://www.ti.com/lit/an/slaa338/slaa338.pdf.

[16] TX6001, RX6001 datasheets. http://www.rfm.com.

[17] W16 Wireless Recording Device. http://www.multichannelsystems.com/products/wireless-systems.

[18] Confidentiality Letter Medtronic. https://apps.fcc.gov/eas/GetApplicationAttachment.html?id=1755518, 2012.

[19] Pacemaker Ratients at Risk of Death by Hacking. http://www.tirosec.com/pacemaker-patients-at-risk-of-death-by-hacking/, 2012.

[20] Doctors Disabled Wireless in Dick Cheney's Pacemaker to Thwart Hacking. https://nakedsecurity.sophos.com/2013/10/22/doctors-disabled-wireless-in-dick-cheneys-pacemaker-to-thwart-hacking/, 2013.

[21] Bitglass Healthcare Breach Report. https://pages.bitglass.com/pr-2014-healthcare-breach-report.html, 2014.

[22] FDA Cybersecurity. https://www.fda.gov/MedicalDevices/DigitalHealth/ucm373213.htm, 2014.

[23] Global Report on Diabetes. http://apps.who.int/iris/bitstream/10665/204871/1/9789241565257_eng.pdf, 2014.

[24] WiFi System Detects Peoples Breathing Heart Rate even Through Walls. https://www.medgadget.com/2014/06/mits-wifi-system-detects-peoples-breathing-heart-rate-even-through-walls.html, 2014.

[25] 32 Dollar RollJam Device can Break Into Most Dars and Garage Doors. https://www.csoonline.com/article/2968312/microsoft-subnet/32-rolljam-device-can-break-into-most-cars-and-garage-doors.html, 2015.

[26] FDA Warns of Security Flaw in Hospira Infusion Pumps. https://www.reuters.com/article/us-hospira-fda-cybersecurity/fda-warns-of-security-flaw-in-hospira-infusion-pumps-idUSKCN0Q52GJ20150731, 2015.

[27] Symbiq Infusion System by Hospira: FDA Safety Communication - Cybersecurity Vulnerabilities. https://www.fda.gov/Safety/MedWatch/SafetyInformation/SafetyAlertsforHumanMedicalProducts/ucm456832.htm, 2015.

[28] Hacking Report on St. Jude Pacemakers Was Flawed, Researchers Say. http://fortune.com/2016/08/31/hacking-st-jude-pacemakers-flawed/, 2016.

[29] Muddy Waters Research on St Jude Medical hacks. http://www.muddywatersresearch.com/research/stj/mw-is-short-stj/, 2016.

[30] St Jude Sues Short-selling MedSec over Pacemaker Hack Report. https://www.theregister.co.uk/2016/09/07/st_jude_sues_over_hacking_claim/, 2016.

[31] A Brief Chronology of Medical Device Security. https://cacm.acm.org/magazines/2016/10/207766-a-brief-chronology-of-medical-device-security/fulltext, 2017.

[32] FDA Issues Recall of 465,000 St. Jude Pacemakers to Patch Security Holes. http://www.zdnet.com/article/fda-forces-st-jude-pacemaker-recall-to-patch-security-vulnerabilities/, 2017.

[33] Global Pacemakers and Implantable Cardioverter Defibrillators (ICDs) Market Analysis and Forecasts 2017-2023. https://globenewswire.com/news-release/2017/06/22/1027754/0/en/Global-Pacemakers-and-Implantable-Cardioverter-Defibrillators-ICDs-Market-Analysis-and-Forecasts-2017-2023.html, 2017.

[34] Pacemakers Fail More Often Than Manufacturers Acknowledge. http://www.pbs.org/wgbh/nova/next/body/pacemakers-fail-more-often-than-manufacturers-acknowledge/, 2017.

[35] Martín Abadi and Cédric Fournet. Mobile Values, New Names, and Secure Communication. In *Proceedings of the 28th SIGPLAN Symposium on Principles of Programming Languages*, POPL, pages 104–115, NY, USA, 2001.

[36] Carlisle Adams and Steve Lloyd. *Understanding PKI: Concepts, Standards, and Deployment Considerations*. Addison-Wesley Longman Publishing Co., Inc., Boston, USA, 2nd edition, 2002.

[37] Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. The EM Side-Channel(s). In *Proceedings of the 4th International*

Workshop on Cryptographic Hardware and Embedded Systems, CHES, pages 29–45, London, UK, 2003. Springer-Verlag.

[38] S. Abhishek Anand and Nitesh Saxena. Vibreaker: Securing Vibrational Pairing with Deliberate Acoustic Noise. In *Proceedings of the 9th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec, pages 103–108, NY, USA, 2016. ACM.

[39] S Abhishek Anand and Nitesh Saxena. Coresident Evil: Noisy Vibrational Pairing in the Face of Co-located Acoustic Eavesdropping. In *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec, pages 173–183, NY, USA, 2017. ACM.

[40] Priyanka Bagade, Ayan Banerjee, Joseph Milazzo, and Sandeep K.S. Gupta. Protect your BSN: No Handshakes, just Namaste! In *IEEE International Conference on Body Sensor Networks*, pages 1–6, 2013.

[41] Constantine A. Balanis. *Antenna Theory: Analysis and Design*. Wiley-Interscience, 2005.

[42] Elaine Barker and John Kelsey. Recommendation for the Entropy Sources Used for Random BitGeneration. NIST DRAFT Special Publication 800-90B, 2016.

[43] Bruno Blanchet, Ben Smyth, and Vincent Cheval. ProVerif 1.88: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial, 2013.

[44] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short Group Signatures. In *Proceedings of the 24th Annual International Cryptology Conference on Advances in Cryptology*, volume 3152 of *CRYPTO*, pages 41–55, Berlin, Heidelberg, 2004.

[45] R.C. Bose and D.K. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3(1):68 – 79, 1960.

[46] Stefan Brands and David Chaum. Distance-bounding protocols. In *Workshop on the Theory and Application of Cryptographic Techniques on Advances in Cryptology*, EUROCRYPT, pages 344–359, Secaucus, NJ, USA, 1994. Springer-Verlag New York, Inc.

[47] Alejandro Calleja, Pedro Peris-Lopez, and Juan E. Tapiador. *Electrical Heart Signals can be Monitored from the Moon: Security Implications for IPI-Based Protocols*, pages 36–51. WISTP. 2015.

[48] Claude Castelluccia and Pars Mutaf. Shake Them Up!: A Movement-based Pairing Protocol for CPU-constrained Devices. In *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services*, MobiSys, pages 51–64, NY, USA, 2005.

[49] Sriram Cherukuri, Krishna K Venkatasubramanian, and Sandeep K S Gupta. Biosec: a biometric based approach for securing communication in wireless networks of biosensors implanted in the human body. In *Proceedings of the International Conference on Parallel Processing Workshops*, pages 432–439, 2003.

[50] Humphrey Cheung. How to: Building a bluesniper rifle - part 1. `http://www.tomsguide.com/us/how-to-bluesniper-pt1,review-408.html`, 2005.

[51] Weidong Cui, Jayanthkumar Kannan, and Helen J. Wang. Discoverer: Automatic Protocol Reverse Engineering from Network Traces. In *Proceedings of the 16th USENIX Security Symposium*, pages 14:1–14:14, Berkeley, CA, USA, 2007.

[52] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[53] Joep de Groot, Boris Škorić, Niels de Vreede, and Jean-Paul Linnartz. Quantization in zero leakage helper data schemes. *EURASIP Journal on Advances in Signal Processing*, 2016(1):54, 2016.

[54] Tamara Denning, Kevin Fu, and Tadayoshi Kohno. Absence Makes the Heart Grow Fonder: New Directions for Implantable Medical Device Security. In *Proceedings of the 3rd Conference on Hot Topics in Security*, HOTSEC, pages 5:1–5:7, Berkeley, USA, 2008. USENIX Association.

[55] Tamara Denning, Daniel B. Kramer, Batya Friedman, Matthew R. Reynolds, Brian Gill, and Tadayoshi Kohno. CPS: Beyond Usability: Applying Value Sensitive Design Based Methods to Investigate Domain Characteristics for Security for Implantable Cardiac Devices. In *Proceedings of the 30th Annual Computer Security Applications Conference*, ACSAC, pages 426–435. ACM, 2014.

[56] Tamara Denning, Yoky Matsuoka, and Tadayoshi Kohno. Neurosecurity: security and privacy for neural devices. *Neurosurgical Focus*, 27(1), 2009.

[57] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Proceedings of the Annual International Conference on the Theory and*

Applications of Cryptographic Techniques, EUROCRYPT, pages 523–540, 2004.

[58] Chris Eagle. *The IDA Pro Book: The Unofficial Guide to the World's Most Popular Disassembler*. No Starch Press, San Francisco, CA, USA, 2008.

[59] Francis F. Chen. *Capacitor tuning circuits for inductive loads*. PPG-1401, 1992.

[60] Bishop Fox. Rfid hacking tools. `https://www.bishopfox.com/resources/tools/rfid-hacking/attack-tools/`.

[61] Great Scott Gadgets. Great Scott Gadgets - HackRF One. `https://greatscottgadgets.com/hackrf/`.

[62] Great Scott Gadgets. Great Scott Gadgets - YARD Stick One. `https://greatscottgadgets.com/yardstickone/`.

[63] Flavio D. Garcia, Gerhard de Koning Gans, Roel Verdult, and Milosch Meriac. Dismantling iClass and iClass Elite. In *Proceedings of the 17th European Symposium on Research in Computer Security*, ESORICS, pages 697–715, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[64] Flavio D. Garcia, Gerhard Koning Gans, Ruben Muijrers, Peter Rossum, Roel Verdult, Ronny Wichers Schreur, and Bart Jacobs. Dismantling MIFARE Classic. In *Proceedings of the 13th European Symposium on Research in Computer Security*, ESORICS, pages 97–114, 2008.

[65] Flavio D. Garcia, David Oswald, Timo Kasper, and Pierre Pavlidès. Lock It and Still Lose It - on the (In)Security of Automotive Remote Keyless Entry Systems. In *Proceedings of the 25th USENIX Security Symposium*, Austin, TX, 2016. USENIX Association.

[66] Christian Gehrmann, Chris J Mitchell, and Kaisa Nyberg. Manual authentication for wireless devices. *RSA Cryptobytes*, 7(1):29–37, 2004.

[67] Shyamnath Gollakota, Haitham Hassanieh, Benjamin Ransford, Dina Katabi, and Kevin Fu. They Can Hear Your Heartbeats: Non-invasive Security for Implantable Medical Devices. *SIGCOMM Computer Communication.*, 41(4):2–13, August 2011.

[68] Tzipora Halevi and Nitesh Saxena. On pairing constrained wireless devices based on secrecy of auxiliary channels: the case of acoustic eavesdropping. In *Proceedings of the 17th Conference on Computer and Communications Security*, CCS, pages 97–108, 2010.

[69] Daniel Halperin, Thomas S. Heydt-Benjamin, Kevin Fu, Tadayoshi Kohno, and William H. Maisel. Security and Privacy for Implantable Medical Devices. *IEEE Pervasive Computing, Special Issue on Implantable Electronics*, 7(1):30–39, 2008.

[70] Daniel Halperin, Thomas S. Heydt-Benjamin, Benjamin Ransford, Shane S. Clark, Benessa Defend, Will Morgan, Kevin Fu, Tadayoshi Kohno, and William H. Maisel. Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In *Proceedings of the 29th IEEE Symposium on Security and Privacy*, SP, pages 129–142, 2008.

[71] Gerhard P. Hancke and Markus G. Kuhn. An RFID Distance Bounding Protocol. In *Proceedings of the 1th International Conference on Security and Privacy for Emerging Areas in Communications Networks*, SECURECOMM, pages 67–73, Washington, DC, USA, 2005. IEEE Computer Society.

[72] Xiali Hei, Xiaojiang Du, Shan Lin, and Insup Lee. PIPAC: Patient infusion pattern based access control scheme for wireless insulin pump system. In *Proceedings of the IEEE International Conference on Computer Communications*, INFOCOM, pages 3030–3038, 2013.

[73] Xiali Hei, Xiaojiang Du, Jie Wu, and Fei Hu. Defending Resource Depletion Attacks on Implantable Medical Devices. In *Global Telecommunications Conference*, GLOBECOM, pages 1–5, 2010.

[74] Josef Hlavác, Róbert Lórencz, and Martin Hadácek. True random number generation on an Atmel AVR microcontroller. In *Proceedings of the 2nd International Conference on Computer Engineering and Technology*, volume 2, pages V2–493–V2–495, 2010.

[75] A. Hocquenghem. Codes Correcteurs d'Erreurs. *Chiffres (Paris)*, 2:147–156, September 1959.

[76] Daniel E. Holcomb, Wayne P. Burleson, and Kevin Fu. Power-Up SRAM State As an Identifying Fingerprint and Source of True Random Numbers. *IEEE Transactions on Computers*, 58(9):1198–1210, September 2009.

[77] Chunqiang Hu, Xiuzhen Cheng, Fan Zhang, Dengyuan Wu, Xiaofeng Liao, and Dechang Chen. OPFKA: Secure and efficient Ordered-Physiological-Feature-based key agreement for wireless Body Area Networks. In *Proceedings of the IEEE International Conference on Computer Communications*, INFOCOM, pages 2274–2282, 2013.

[78] Steven J. Murdoch, Saar Drimer, Ross Anderson, and Mike Bond. Chip and PIN is Broken. In *Proceedings of the IEEE Symposium on Security and Privacy*, SP, pages 433–446, 2010.

[79] Don Johnson, Alfred Menezes, and Scott Vanstone. The Elliptic Curve Digital Signature Algorithm. *In Prooceding of the International Journal of Information Security*, 1(1):36–63, 2014.

[80] Younghyun Kim, Woo Suk Lee, Vijay Raghunathan, Niraj K. Jha, and Anand Raghunathan. Vibration-based Secure Side Channel for Medical Devices. In *Proceedings of the 52nd Annual Design Automation Conference*, DAC, pages 32:1–32:6, NY, USA, 2015. ACM.

[81] Jeonggil Ko, Jong Hyun Lim, Yin Chen, Rvăzvan Musvaloiu-E, Andreas Terzis, Gerald M. Masson, Tia Gao, Walt Destler, Leo Selavo, and Richard P. Dutton. MEDiSN: Medical Emergency Detection in Sensor Networks. *ACM Transactions on Embedded Computing Systems*, 10(1):11:1–11:29, 2010.

[82] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO, pages 388–397, London, UK, 1999.

[83] Philip Koopman and Tridib Chakravarty. Cyclic redundancy code (CRC) polynomial selection for embedded networks. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 145–154, 2004.

[84] Denis Foo Kune, John D. Backes, Shane S. Clark, Daniel B. Kramer, Matthew R. Reynolds, Kevin Fu, Yongdae Kim, and Wenyuan Xu. Ghost Talk: Mitigating EMI Signal Injection Attacks against Analog Sensors. In *Proocedings of the IEEE Symposium on Security and Privacy*, pages 145–159, Berkeley, USA, 2013.

[85] Nancy. G. Leveson and Clark. S. Turner. An Investigation of the Therac-25 Accidents. *Computer*, 26(7):18–41, July 1993.

[86] Chunxiao Li, Anand Raghunathan, and Niraj K. Jha. Hijacking an insulin pump: Security attacks and defenses for a diabetes therapy system. In *Proceedings of the 13th International Conference on e-Health Networking Applications and Services*, pages 150–156, 2011.

[87] Jean-Paul Linnartz and Pim Tuyls. New Shielding Functions to Enhance Privacy and Prevent Misuse of Biometric Templates. In *Proceedings of the 4th International Conference on Audio- and Video-based Biometric Person Authentication*, AVBPA, pages 393–402, Berlin, Heidelberg, 2003. Springer-Verlag.

[88] David Malan, Fulford-Jones Thaddeus, Matt Welsh, and Steve Moulton. CodeBlue: An Ad Hoc Sensor Network Infrastructure for Emergency Medical Care. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services*, MobiSys, pages 12–14. ACM, 2004.

[89] Eduard Marin, Enrique Argones Rúa, Dave Singelée, and Bart Preneel. A survey on physiological-signal-based security for medical devices. Cryptology ePrint Archive, Report 2016/188, 2016.

[90] Eduard Marin, Mustafa A. Mustafa, Dave Singelée, and Bart Preneel. A privacy-preserving remote healthcare system offering end-to-end security. In *Proceedings of the 15th International Conference on Ad-Hoc Networks and Wireless*, ADHOC-NOW, pages 237–250, 2016.

[91] Eduard Marin, Dave Singelée, Flavio D. Garcia, Tom Chothia, Rik Willems, and Bart Preneel. On the (in)Security of the Latest Generation Implantable Cardiac Defibrillators and How to Secure Them. In *Proceedings of the 32th Annual Conference on Computer Security Applications*, ACSAC, pages 226–236, 2016.

[92] Eduard Marin, Dave Singelée, Bohan Yang, Ingrid Verbauwhede, and Bart Preneel. On the Feasibility of Cryptography for a Wireless Insulin Pump System. In *Proceedings of the 6th Conference on Data and Application Security and Privacy*, CODASPY, pages 113–120, 2016.

[93] Eduard Marin, Dave Singelée, Bohan Yang, Vladimir Volski, Guy A. E. Vandenbosch, Bart Nuttin, and Bart Preneel. Securing Wireless Neurostimulators. In *Proceedings of the 8th Conference on Data and Application Security and Privacy (to appear)*, CODASPY, 2018.

[94] Ivan Martinovic, Doug Davies, Mario Frank, Daniele Perito, Tomas Ros, and Dawn Song. On the Feasibility of Side-channel Attacks with Brain-computer Interfaces. In *Proceedings of the 21st USENIX Conference on Security Symposium*, pages 34–34, 2012.

[95] Gerlinde A Metz and Ian Q Whishaw. The ladder rung walking task: a scoring system and its practical application. *Journal of Visualized Experiments*, (28):e1204–e1204, 2009.

[96] Fen Miao, Shu-Di Bao, and Ye Li. A modified fuzzy vault scheme for biometrics-based body sensor networks security. In *Proceedings of the Global Communications Conference*, GLOBECOM, pages 1–5, Miami, USA, 2010.

[97] Dmitry Nedospasov. NXP LPC1343 Bootloader Bypass (Part 1) - Communicating with the bootloader. `https://toothless.co/blog/bootloader-bypass-part1/`, 2017.

[98] Jason W. P. Ng, Benny P. L. Lo, Oliver Wells, Morris Sloman, Nick Peters, Ara Darzi, Chris Toumazou, and Guang Z. Yang. Ubiquitous Monitoring Environment for Wearable and Implantable Sensors. In *Proceedings of the 6th International Conference on Ubiquitous Computing*, UbiComp. 2004.

[99] Nuand. Nuand | bladeRF Software Defined Radio. `https://www.nuand.com/`.

[100] Andres Ortiz, Jorge Munilla, and Alberto Peinado. Secure wireless data link for low-cost telemetry and telecommand applications. In *Proceedings of the IEEE Mediterranean Electrotechnical Conference*, MELECON, pages 828–831. IEEE, 2006.

[101] Lara Ortiz-Martin, Pablo Picazo-Sanchez, Pedro Peris-Lopez, and Juan Tapiador. Heartbeats Do Not Make Good Pseudo-Random Number Generators: An Analysis of the Randomness of Inter-Pulse Intervals. *Entropy*, 20(2), 2018.

[102] Adrian Perrig, Robert Szewczyk, J. D. Tygar, Victor Wen, and David E. Culler. SPINS: Security Protocols for Sensor Networks. *Wireless Networks*, 8(5):521–534, September 2002.

[103] Ming-Zher Poh, Daniel J. McDuff, and Rosalind W. Picard. Advancements in Noncontact, Multiparameter Physiological Measurements Using a Webcam. *IEEE Transactions on Biomedical Engineering*, 58(1):7–11, Jan 2011.

[104] C. C. Y. Poon, Yuan-Ting Zhang, and Shu-Di Bao. A novel biometrics method to secure wireless body area sensor networks for telemedicine and m-health. *IEEE Communications Magazine*, 44(4):73–81, April 2006.

[105] John. Proakis and Masoud Salehi. *Digital Communications*. McGraw-Hill higher education. McGraw-Hill Education, 2007.

[106] Jay Radcliffe. Hacking Insulin Pumps for Fun and Insulin: Breaking the Human SCADA System. `https://media.blackhat.com/bh-us-11/Radcliffe/BH_US_11_Radcliffe_Hacking_Medical_Devices_Slides.pdf`, 2011.

[107] Jay Radcliffe. R7-2016-07: Multiple Vulnerabilities in Animas OneTouch Ping Insulin Pump. `https://blog.rapid7.com/2016/10/04/r7-2016-07-multiple-vulnerabilities-in-animas-onetouch-ping-insulin-pump/`, 2016.

[108] Aanjhan Ranganathan, Nils Ole Tippenhauer, Boris Škorić, Dave Singelée, and Srdjan Čapkun. Design and implementation of a terrorist fraud

resilient distance bounding system. In Sara Foresti, Moti Yung, and Fabio Martinelli, editors, *Proceedings of the 17th European Symposium on Research in Computer Security*, ESORICS, pages 415–432, 2012.

[109] Kasper Bonne Rasmussen, Claude Castelluccia, Thomas S. Heydt-Benjamin, and Srdjan Capkun. Proximity-based Access Control for Implantable Medical Devices. In *Proceedings of the 16th Conference on Computer and Communications Security*, CCS, pages 410–419, 2009.

[110] Kasper Bonne Rasmussen and Srdjan Čapkun. Realization of RF Distance Bounding. In *Proceedings of the 19th USENIX Conference on Security*, USENIX Security, pages 25–25, Berkeley, CA, USA, 2010. USENIX Association.

[111] Luca Reverberi and David Oswald. Breaking (and fixing) a widely used continuous glucose monitoring system. In *Proocedings of the 11th Workshop on Offensive Technologies*, WOOT, Vancouver, BC, 2017. USENIX Association.

[112] Billy Rios and Jonathan Butts. Security Evaluation of the Implantable Cardiac Device Ecosystem Architecture and Implementation Interdependencies. `https://www.ledecodeur.ch/wp-content/uploads/2017/05/Pacemaker-Ecosystem-Evaluation.pdf`, 2017.

[113] Masoud Rostami, Wayne Burleson, Farinaz Koushanfar, and Ari Juels. Balancing security and utility in medical devices? In *Proocedings of the 50th Design Automation Conference*, DAC, pages 13:1–13:6, 2013.

[114] Masoud Rostami, Ari Juels, and Farinaz Koushanfar. Heart-to-Heart (H2H): Authentication for Implanted Medical Devices. In *Proceedings of the 20th Computer and Communications Security* [114], pages 1099–1112.

[115] Ishtiaq Rouf, Hossen Mustafa, Miao Xu, Wenyuan Xu, Rob Miller, and Marco Gruteser. Neighborhood Watch: Security and Privacy Analysis of Automatic Meter Reading Systems. In *Proceedings of the Conference on Computer and Communications Security*, CCS, pages 462–473, NY, USA, 2012. ACM.

[116] Michael Rushanan, Aviel D. Rubin, Denis Foo Kune, and Colleen M. Swanson. SoK: Security and Privacy in Implantable Medical Devices and Body Area Networks. In *Proceedings of the IEEE Symposium on Security and Privacy*, SP, pages 524–539, 2014.

[117] Huseyin S. Savci, Ahmet Sula, Zheng Wang, and Numan S. Dogan. MICS transceivers: regulatory standards and applications. In *SoutheastCon. Proc. IEEE*, 2005.

[118] Krishna Sampigethaya and Radha Poovendran. A Survey on Mix Networks and Their Secure Applications. *Proceedings of the IEEE*, 94(12), 2006.

[119] Nitesh Saxena, Md. Borhan Uddin, Jonathan Voris, and N. Asokan. Vibrate-to-unlock: Mobile phone assisted user authentication to multiple personal RFID tags. In *IEEE International Conference on Pervasive Computing and Communications*, PerCom, pages 181–188, 2011.

[120] Sahar Sedighpour, Srdjan Capkun, Saurabh Ganeriwal, and Mani Srivastava. Implementation of attacks on ultrasonic ranging systems, 2005.

[121] Robert M. Seepers, Christos Strydis, Ioannis Sourdis, and Chris I. De Zeeuw. On Using a Von Neumann Extractor in Heart-Beat-Based Security. In *IEEE Trustcom/BigDataSE/ISPA*, volume 1, pages 491–498, Aug 2015.

[122] Robert M. Seepers, Wenjin Wang, Gerard de Haan, Ioannis Sourdis, and Christos Strydis. Attacks on Heartbeat-Based Security Using Remote Photoplethysmography. *IEEE Journal of Biomedical and Health Informatics*, PP(99):1–1, 2017.

[123] Dave Singelée, Stefaan Seys, Lejla Batina, and Ingrid Verbauwhede. The energy budget for wireless security: Extended version. *IACR Cryptology ePrint Archive*, 2015:1029, 2015.

[124] Dave Singelee, Stefaan Seys, Lejla Batina, and Ingrid Verbauwhede. The Communication and Computation Cost of Wireless Security: Extended Abstract. In *Proceedings of the 4th ACM Conference on Wireless Network Security*, WiSec, pages 1–4, NY, USA, 2011. ACM.

[125] Sergei Skorobogatov. Deep dip teardown of tubeless insulin pump. *CoRR*, abs/1709.06026, 2017.

[126] Matthew Smith, Daniel Moser, Martin Strohmeier, Vincent Lenders, and Ivan Martinovic. Economy Class Crypto: Exploring Weak Cipher Usage in Avionic Communications via ACARS. In *Proceedings of the 21st International Conference on Financial Cryptography and Data Security*, FC, 2017.

[127] Frank Stajano and Ross J. Anderson. The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In *Proceedings of the 7th International Workshop on Security Protocols*, pages 172–194, 2000.

[128] Nils Ole Tippenhauer, Lula Malisa, Aanjhan Ranganathan, and Srdjan Capkun. On Limitations of Friendly Jamming for Confidentiality. In

Proceedings of the IEEE Symposium on Security and Privacy, SP, pages 160–173, 2013.

[129] Timothy Trippel, Ofir Weisse, Wenyuan Xu, Peter Honeyman, and Kevin Fu. WALNUT: Waging Doubt on the Integrity of MEMS Accelerometers with Acoustic Injection Attacks. In *Proceedings of the 2nd European Symposium on Security and Privacy*, EuroSP, 2017.

[130] Anthony Van Herrewege, Vincent van der Leest, André Schaller, Stefan Katzenbeisser, and Ingrid Verbauwhede. Secure PRNG Seeding on Commercial Off-the-shelf Microcontrollers. In *Proceedings of the 3rd International Workshop on Trustworthy Embedded Devices*, TrustED, pages 55–64, NY, USA, 2013. ACM.

[131] Krishna K Venkatasubramanian, Ayan Banerjee, and S Gupta. Plethysmogram-based secure inter-sensor communication in body area networks. In *Proceedings of the IEEE Conference on Military Communications*, MILCOM, pages 1–7. IEEE, 2008.

[132] Krishna K. Venkatasubramanian, Ayan Banerjee, and Sandeep Kumar S. Gupta. PSKA: Usable and Secure Key Agreement Scheme for Body Area Networks. *IEEE Transactions on Information Technology in Biomedicine*, 14(1):60–68, January 2010.

[133] Doug Whiting, Russ Housley, and Niels Ferguson. Counter with CBC-MAC (CCM). RFC 3610, September 2003.

[134] Fengyuan Xu, Zhengrui Qin, Chiu Chiang Tan, Baosheng Wang, and Qun Li. IMDGuard: Securing implantable medical devices with the external wearable guardian. In *Proceedings of the 30th International Conference on Computer Communications*, pages 1862–1870, 2011.

[135] Lin Yao, Bing Liu, Guowei Wu, Kai Yao, and Jia Wang. A Biometric Key Establishment Protocol for Body Area Networks. *IJDSN*, 2011, 2011.

[136] Meng Zhang, Anand Raghunathan, and Niraj K. Jha. MedMon: Securing Medical Devices Through Wireless Monitoring and Anomaly Detection. *IEEE Transactions on Biomedical Circuits and Systems*, 7(6):871–881, 2013.

Curriculum vitae

Eduard Marin was born on October 15, 1989 in Barcelona, Spain. He received the Bachelor's degree in Telecommunications Systems Engineering (Ingeniero Técnico de Telecomunicación, especialidad en Sistemas de Telecomunicación) from Universitat Politécnica de Catalunya (UPC) in 2011. In 2013, he obtained the Master's degree in Telecommunications Engineering (Ingeniero Superior de Telecomunicaciones) from Universitat Politécnica de Catalunya (UPC) Engineering. His Master's thesis was done as Erasmus student at the research group COSIC (Computer Security and Industrial Cryptography) at the Department of Electrical Engineering (ESAT) of the KU Leuven. In October 2013 he joined COSIC as a PhD student under the supervision of Prof. Bart Preneel and Dr. Dave Singelée.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
COSIC
Kasteelpark Arenberg 10, bus 2452
B-3001 Leuven
eduard.marin@esat.kuleuven.be
https://securewww.esat.kuleuven.be/cosic/